IMPROVING STATISTICAL MACHINE TRANSLATION THROUGH *N*-BEST LIST RE-RANKING AND OPTIMIZATION

THESIS

Jordan S. Keefer, Second Lieutenant, USAF

AFIT-ENG-14-M-43

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

DISTRIBUTION STATEMENT A:
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

The views expressed in this thesis are those of the author and do not reflect the official policy or position of the United States Air Force, the Department of Defense, or the United States Government.

This material is declared a work of the U.S. Government and is not subject to copyright protection in the United States.

AFIT-ENG-14-M-43

IMPROVING STATISTICAL MACHINE TRANSLATION THROUGH *N*-BEST LIST RE-RANKING AND OPTIMIZATION

THESIS

Presented to the Faculty

Department of Electrical and Computer Engineering

Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

Air Education and Training Command

in Partial Fulfillment of the Requirements for the

Degree of Master of Science in Cyber Operations

Jordan S. Keefer, B.S.C.S.

Second Lieutenant, USAF

March 2014

DISTRIBUTION STATEMENT A:
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

AFIT-ENG-14-M-43

IMPROVING STATISTICAL MACHINE TRANSLATION THROUGH *N*-BEST LIST RE-RANKING AND OPTIMIZATION

Jordan S. Keefer, B.S.C.S.
Second Lieutenant, USAF

Approved:

//signed//	12 Mar 2014
Maj Kennard R. Laviers, PhD (Chairman)	Date
//signed//	13 Mar 2014
Timothy R. Anderson, PhD (Member)	Date
//signed//	10 Mar 2014
Kenneth M. Hopkinson, PhD (Member)	Date
//signed//	12 Mar 2014
Gilbert L. Peterson, PhD (Member)	Date

AFIT-ENG-14-M-43

Abstract

Statistical machine translation (SMT) is a method of translating from one natural language (NL) to another using statistical models generated from examples of the NLs. The quality of translation generated by SMT systems is competitive with other premiere machine translation (MT) systems and more improvements can be made. This thesis focuses on improving the quality of translation by re-ranking the *n*-best lists that are generated by modern phrase-based SMT systems. The *n*-best lists represent the *n* most likely translations of a sentence. The research establishes upper and lower limits of the translation quality achievable through re-ranking. Three methods of generating an *n*-gram language model (LM) from the *n*-best lists are proposed. Applying the LMs to re-ranking the *n*-best lists results in improvements of up to six percent in the Bi-Lingual Evaluation Understudy (BLEU) score of the translation.

Table of Contents

Page

List of Figures

List of Tables

List of Acronyms

Acronym	Definition
AFRL	Air Force Research Laboratory
ALPAC	Automatic Language Processing Advisory Committee
AI	artificial intelligence
BFS	breadth-first search
BLEU	bilingual evaluation understudy
BRICS	Brazil, Russia, India, China, and South Africa
CFG	context-free grammar
DoD	Department of Defense
EM	Expectation Maximization
EU	European Union
FTD	Foreign Translation Division
IBM	International Business Machine
ICAO	International Civil Aviation Organization
IMO	International Maritime Organization
IWSLT	International Workshop on Spoken Language Translation
LM	language model
MERT	Minimum Error Rate Training
MITLL	Massachusetts Institute of Technology Lincoln Laboratory
MT	machine translation
NATO	North Atlantic Trade Organization
NIST	National Institute of Standards and Technology
NL	natural language
NSF	National Science Foundation

Acronym	Definition
RBMT	rule-based machine translation
SCFG	synchronous context-free grammar
SGE	Sun Grid Engine
SL	source language
SMT	statistical machine translation
SRILM	SRI Language Model
TED	Technology, Entertainment, Design
TL	target language
TM	translation model
UN	United Nations
USAF	United States Air Force
WIT^3	Web Inventory of Transcribed and Translated Talks
WPAFB	Wright-Patterson Air Force Base
XML	Extensible Markup Language
Bash	Bourne-Again Shell

IMPROVING STATISTICAL MACHINE TRANSLATION THROUGH *N*-BEST LIST RE-RANKING AND OPTIMIZATION

I. Introduction

In an increasingly global society, the ability to communicate across multiple languages is invaluable. Large and small nations are interconnected. From ancient civilizations like China to the newly recognized South Sudan, countries are interacting and joining large multicultural treaties, pacts, and trade agreements such as the United Nations (UN), European Union (EU), North Atlantic Trade Organization (NATO), and Brazil, Russia, India, China, and South Africa (BRICS). Military imperialism has given way to global corporations with offices in multiple countries.

Many of these changes are impacts of technology. Technology makes travel easy with personal automobiles, multiple airlines, and high speed rail networks that span continents [10]. In recent decades, these forms of transit have become mainstream, bringing people closer together. The internet, now a mainstay of modern culture and society, is the embodiment of information and technology [31]. More technological advances, such as a fiber-optic backbone and remote satellite links, make it available in almost every corner of the globe. Video and textual chat clients allow people to communicate globally for free [3]. Now information can be shared instantly and at low cost. Scientific advances made in one country are shared with the world in a matter of seconds. Citizens and leaders around the world can directly communicate over the internet. And yet for all the physical and digital connectivity, there still remains an immense barrier that separates and limits peoples' ability to communicate, interact, and collaborate. This is the language barrier, and while technology enables humanity to overcome many other barriers facing society, the language gap still represents a formidable challenge.

Traditionally, the only way to bridge a language gap is to have a translator. At the time of this writing, there are over 7,000 living languages with 22 having over 50 million primary language speakers [15]. These numbers make relying on fluency to avoid language gaps or employing human translators to bridge them infeasible. An individual or group usually requires translations between only a subset of these world languages. Even so, bridging the language gap is expensive and difficult [16]. The UN identifies Arabic, Chinese, English, French, Russian and Spanish as its official languages [14]. This means that a member may speak in any of these languages with the expectation of being understood and that what they said will be translated into all the other languages [54]. The UN may be able to employ individuals to translate its official languages, but there is a reason the official languages are limited. The problem becomes insurmountable if the language barrier is between two private individuals without access to a translator. The individuals must resort to a bilingual dictionary or pictures to communicate simple ideas. The problem is equally blaring for organizations such as the military who do not have the luxury of limiting their range of foreign language involvement.

International organizations sometimes establish a language treaty for their members.For example, the International Maritime Organization (IMO) establishes a phrase book of mandatory phrases that the individuals governed by the IMO should use. Another example is the International Civil Aviation Organization (ICAO), which establishes English as the language for aviation radio communication and requires countries' aviation organizations to ensure their members are proficient in English [8].

Treatise work in organizations with members that consent to abide by the treaty. Treatise between entities are different however. In the general case such agreements are faulty. It is not considered a good diplomatic practice for a nation to try to establish its language as the world language. Even if its language spreads innocently through natural or diplomatic means, it is referred to as linguistic imperialism [62]. Additionally people are usually less open when approached with a non-native language. In fact, many people only understand their native tongue. In global application such as multi-national coalitions or global disaster relief, where individuals not specifically prepared for foreign interaction may need to be communicated with, it is important to be able to communicate.

In addition to communication, the ability to monitor events around the world is also bolstered by translation. With many news sources making their content available online, all that remains is to translate it into a familiar language. The volume of news available in this format makes it infeasible to manually translating everything in order to find something of interest. Having even a rough translation would be better, allowing individuals to get the gist of the information and select which articles they want translated by humans.

The solution technology offers to these problems is machine translation (MT). It has the potential to be an available, fast, and tireless translator. Making a machine translate is not easy. While there are computers processing language, these devices often fill the roles of word processors, filing cabinets, or messengers. The computers do not need to understand what they are processing. Computers are weak at tasks that require human qualities like understanding and innovation [78]. These are fundamental skills in the understanding of a natural language (NL) and consequentially translation [63]. NLs are used by societies and are forged over time to enable people to communicate about their environment and their thoughts. They are therefore as varied and vast as the different people who use them. In addition NLs can be as flexible and ambiguous as the world they were created to describe. Thus, applying the discrete logic of computers to the task of translation is very difficult and not yet fully understood.

1.1 Problem Description

It is important to understand that while the golden standard of translation remains human translation, it is not perfect. The quality of a translation is objectively the degree with which the translation holds the same meaning as the originator intended to relay [63]. A human translator does not simply replace each source word in one language with the corresponding word in the other language. When a human translator receives the source language (SL), he or she first understands by interpreting the message and synthesizing the meaning intended by the originator. The translator does this using context as well as cultural and professional experience. Finally, he or she generates an output in the target language (TL) based on this synthesized meaning and their knowledge of the TL [63]. Different people will choose to encode what they mean to convey in different ways. The result is that even a monolingual conversation can have any number of misunderstandings. For example the phrase “fill the pen” could mean two different things depending on if spoken to a writer or a farmer. This highlights that words can have many different meanings depending on context. Sentence structure can vary, which may or may not change the meaning. Even more ambiguity is caused by the way languages change and adapt over time. The task becomes more complicated when the differences between languages are considered as well. An idea may take a different number of words to represent in different languages. Languages treat references to time differently. For example, an English speaker would say “I am going tomorrow” where an Chinese speaker would say “I go tomorrow.” A language can even have great variance within itself. Dialects and slang are modifications to languages that evolve and vary geographically and temporally. They can change a language so that even a native speaker may have difficulty knowing what is being said. This ambiguity and flexibility is part of what makes MT difficult.

There have been attempts over the years to find a suitable solution to the machine translation problem. In 1964 the Director of the National Science Foundation (NSF), Dr. Leland Haworth, commissioned a research team to generate a report for a number of defense agencies on the current state of research in the field of "mechanical translation of foreign languages." In 1966 the committee published their results in what has come to be know as the Automatic Language Processing Advisory Committee (ALPAC) report [63]. The report identified the fundamentals of translation and defined MT as "going by algorithm from machine-readable source text to useful target text, without recourse to human translation or editing." At the time, most MT systems failed to meet this standard because they relied on human post-editing to fix mistranslations. Systems were usually just for research or to augment the human translation process [63]. The systems that ALPAC reported on were not considered promising by the committee. The committees negative assessment of MT is often blamed for a lull in research [48].

One of the first MT systems to meet mild success was the IBM Mark II, which was a joint project between the International Business Machine (IBM) T.J. Watson Research Center in Yorktown, New York and the United States Air Force (USAF) Foreign Translation Division (FTD) located at Wright-Patterson Air Force Base (WPAFB) in Ohio. The predecessor to this system was the Mark I. Both were described as being similar to a crude electronic bilingual dictionary [63]. Both of these early systems are examples of rule-based machine translation (RBMT) system. They relied on language-specific rules constructed by professionals to make translations. In 1970 the Mark II was replaced by SYSTRAN, which is an example of a contemporary RBMT system and a successful business built around MT [12]. For each new language pair or subject matter, an RBMT system needs rules generated by individuals proficient in both languages. The translation rules are usually extensive and constantly need refining and updating. If a model is purchased to translate medical documents, another model needs to be purchased to translate engineering documents or the systems will produce poor translations [48]. This drives up the cost of purchasing these systems. The algorithms and models used in RBMT systems are usually proprietary and require payment for use. These RBMT systems started to be successful in the late 1970's and early 1980's; however, they were normally contracted to large companies or governments with specific interest in certain language pairs. These organizations were also the ones who could afford the expensive systems and support costs [43].

Early MT systems continued to fight a constant cost-benefit battle against human translators [43]. In 1988 an approach was developed in the IBM T.J. Watson Research Center which proposed the use of a statistical approach to translate language [22]. This approach is called statistical machine translation (SMT), and it is primarily founded on the fact that language is not random, but has a degree of statistically predictable order..

The benefits of SMT systems are numerous. It uses parallel examples of machine-readable SL and TL text to automatically build the translation models, which makes developing MT systems more accessible since training and maintaining SMT systems does not require foreign language skills. They are extensible, and theoretically only need examples of a new language pair or subject matter for them to become proficient. It is therefore cheaper and faster to extend an existing language model or generate a new language pair model than with RBMT. SMT can even be trained for slang, dialects, writing, or speaking because it learns from how language is used. No knowledge of either the SL or TL is needed to build or translate with SMT. Since 2003, SMT is consistently competitive for the top position in translation competitions and is therefore a viable successor to RBMT [46]. Further support of this argument is that Google fully converted their free translation service to SMT in 2009 [4].

1.2 Hypothesis

While SMT systems are becoming more powerful, there are many parts of the SMT process that can still be improved. The supporting hypothesis of this research is that the *n*-gram occurrences within the *n*-best lists can be used to re-rank the *n*-best lists and improve the SMT system's translation. Another important hypothesis is that the scope in which these *n*-grams occur is also important. For example, the *n*-grams that occur in another corpus's *n*-best lists are not expected to be as valuable as the *n*-grams within the corpus. Therefore, the language models should be formed over different scopes which should be likely to have *n*-gram re-occurrences between *n*-best lists. The step in SMT when the system generates the output text is known as decoding. At the end of this stage the system has a list of *n* possible translations for each sentence. Each possible translation is known as a hypothesis. The list of hypotheses is known as the *n*-best list. The list is ordered by a score that is a weighted combination of different sub-scores based on features that are used to estimate the quality of the translation. The best hypothesis from every list is combined to form the translation. Despite optimizations made in this process, the decoding algorithm will often not sort the best translation in the *n*-best list to the top. This is proved using an oracle re-ranking feature, discussed in Chapter 2, which can identify the best hypothesis in an *n*-best list when translating known SL/TL pairs. The *n*-grams should therefore enable one of the better hypotheses to be selected from each *n*-best list to produce a better translation.

1.3 Methodology

This research proposes a process to generate an *n*-gram language model (LM) from the *n*-best lists generated during decoding and then re-rank the *n*-best lists using the LM to produce a better translation. The procedure is done for an entire corpus. The methods used in this research are language-agnostic; however, they are applied only to Arabic to English and German to English translations. Two unique methods of counting the *n*-grams are used. The *n*-gram counts for each *n*-best list are combined over three distinct scopes. The corpus-level combination creates a single LM for the corpus. The producer-level combines the *n*-gram counts based on who generated the source. The local *k*-window uses a window of *k* *n*-gram counts from the surrounding *n*-best lists to generate a LM for each *n*-best list in the corpus. The LMs from the corpus-level and producer-level scopes are applied to re-rank the *n*-best lists from which they were created. The local *k*-window LMs are used to re-rank the focus *n*-best list in the center of the *k* *n*-best lists. The new top hypotheses from the *n*-best lists for the corpus are selected to form a translation which is rated against the known reference translation using the bilingual evaluation understudy (BLEU) metric. The improvement over the BLEU score of the original *n*-best lists are used to measure the improvements made by the system. The BLEU score achieved by a system which made other optimizations on the translation process is used to determine if the quality of the results is significant.

1.4 Overview of Results

Results show that the an improvement of up to six percent in the BLEU score over the translation produced by the phrase based translation system. This improvement is achieved using the corpus-level approach. The scores are comparable to those achieved by a highly optimized system entered into a 2011 translation competition [19]. The producer-level model increase the translation scores around four percent. The lower performance is likely due to the smaller size of training data per language model. The local *k*-window method improved the translation score marginally. This is most likely also due to the small sample data size. Because the *n*-best lists are not available until after decoding, the process cannot be integrated into the decoding algorithm. Generating the *n*-gram LM and re-ranking the *n*-best lists were not computationally complex and were implemented in an interpreted scripting language and executed on a workstation. The process is a viable solution to improving SMT system translations.

II. Background Related work

2.1 Foundation

The concept of Statistical Machine Translation (SMT) was introduced in 1949, stemming from the methods used by the computers in World War II to break codes [77]. When Warren Weaver first presented the concept of SMT he cited Claude Shannon's work in cryptography [67]. He proposed that a language like Russian can be viewed as English encoded in strange characters. He could then proceed with translation the way he would crack a code with statistical approaches. He also believed that all languages were similar enough for this code cracking technique to work because they are all built by humans for the purpose of communicating in essentially the same environment. He also proposed the important concept that the meaning of a word could be statistically determined by viewing *n* words surrounding it [77]. A lack of computing power, digital corpora, and support from academics in the linguistics fields [63] prevented further work for some time. However, in 1988, individuals in IBM's Watson research division revived the idea when they proposed an approach for SMT [24]. This research established many of the principles of the field which remain fundamentally unchanged. Their approach was to use parallel examples of the SL and TL to automatically generate translation rules for language pairs, which are then used to translate unknown SL text.

Once a system is trained it will only translate from SL to TL [25]. To translate the other way one must swap the "sides" of the training material and retrain the system. The SL and TL are typically distinct natural languages like English and Arabic; however, this is not a rule. For example, SMT can be applied to the English results of an RBMT system as a form of automated post editing [68] [35]. Open source SMT software such as Moses [70] has made SMT readily available to researchers and individual users who want to set up an affordable MT solution. In addition, the rise in popularity may have also been bolstered by support of numerous individual and corporate researchers.

2.2 Overview

The exact implementation of SMT varies between systems. Almost every step in the process can be accomplished multiple ways and broken down into sub-steps. Even so, there are a few core steps to SMT which are generally accepted. This overview will cover these foundational concepts and will go through each step of the process while primarily focusing on the specific methods used in this research and prior work which supports it. These methods are referred to as the primary way to accomplish each step.

The first step to understanding SMT, even before understanding the overarching process, is to understand the terminology and how the core elements are discussed. Words are considered the atomic building blocks of languages. This may seem trivial, but before the SMT system is able to translate, the input must be tokenized into the things like punctuation, breaking up contractions, and preserving or normalizing the cases of the words [48]. The tools that handle the tokenization and detokenization (part of preprocessing and post-processing respectively), are well established and separate from the SMT system. For example, the standard distribution of Moses comes with a suite of Perl scripts to handle these processing needs. Words contain a great deal of metadata that hints to a word's meaning. They can be different parts of speech (verb, noun, etc...), capitalized, plural or singular, or a compound word, etc. Even so, it is very difficult to determine the meaning, much less the proper translation, of a word on its own.

A group of contiguous words is referred to as a phrase. The phrases are combined into sentences which are typically the smallest unit of work in translation. This unit of work is known as a segment. A segment is what is taken through the SMT algorithm from start to finish. Sentences are structured and often follow language-specific rules that dictate which parts of speech they contain and where. A sentence typically has a meaning that combines with its neighboring sentences forming an overarching message.

The highest level of organization is the corpus. The corpus is made up of a collection of sentences. The SMT system uses a corpus in training or is given a corpus to translate. Like the previous levels of organization, the corpus has meta-information. It has a domain, such as “medical science” or “politics,” which affects the meanings of words. The corpus’s topic is a clue to which words will be used in it. The corpus will also have a modality, which will affect how the language is structured as well as the quality. There will be a difference in the language used in a typical conversation versus a scholarly article in a psychology journal versus one in a mathematics journal.

As the name implies, SMT uses statistics to accomplish MT. Statistics allow the assignment of mathematical values to describe uncertain events. SMT uses statistics to represent the probability that a given SL segment will translate into a corresponding TL segment. The probability is represented as the function P. The probability of a certain translation is $P(T)$ where T is a TL segment. Translation systems try to maximize the value $P(T)$. Leaving translation generation purely to statistical likelihood would result in every translation being the most common phrase in the TL corpus. Translation is not pure chance; the source sentence S is given as evidence for the translation. The field of mathematics that applies to this type of problem is Bayesian statistics. A full induction into Bayesian statistics is not feasible within the confines of this document; however, a simple example should suffice as an introduction to the unfamiliar.

Statistics speak of a random event x and its probability $P(x)$ that it will occur. The probabilities are determined through observation. For instance, if asking an insurance actuary “Will I be struck by lightning?” the actuary will provide you with the $P(x) = y$ where x is you being struck by lightning and y is the probability that x will happen. The actuary will calculate the probability based on the number of people that get struck versus the population. The actuary may also ask you where you live. This is because lightning strikes are functionally dependent on your location. The actuary not only knows the number of people struck by lightning, but also knows where the strikes occurred. Based on where you live he calculates the probability of x given L which is written as $P(x|L)$ where L is your location.

In SMT, one asks the system the probability of a translation T given the source S. The system returns the probability $P(T|S)$ based on what it observed in the training data. This is represented in Bayes theorem in Equation (2.1).

$$P(T|S) = P(S|T) \cdot \frac{P(T)}{P(S)} \tag{2.1}$$

Bayes theorem allows systems to calculate the probability of T given S. This is useful but it does not solve the problem of generating T. For this Equation (2.2) is derived. This equation is how the system determines the best possible T for the given S and is the foundation of SMT [25].

$$T_{est} = arg\ max_T \frac{P(S|T) \cdot P(T)}{P(S)} = arg\ max_T P(T|S) \cdot P(T) \tag{2.2}$$

The points of knowledge used by the system to give a segment of text a numeric rating are known generally as features. Typical SMT systems will use at least three features: a translation model (TM) that describes $P(T|S)$, a LM that describes how well T statistically conforms to the TL, and the reordering model that gives a cost depending on how much a sentence structure had to be reordered. These features will be discussed in great detail later in this chapter. Feature scores are combined to allow the system to generate the most meaningful and accurate translation.

This is a simple diagram. In the following sections, each component of SMT is examined and explained.

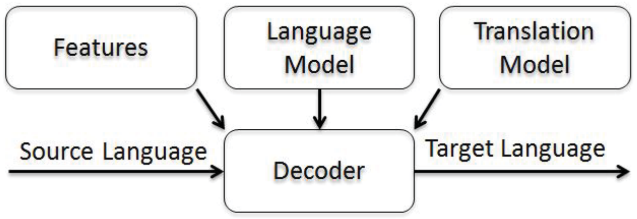


Figure 2.1: Overview of the SMT process.

2.3 Scope

It is important to understand where SMT exists within the scope of the translation process. It is more specific than human translation. Human translators parse input, remove unimportant sounds, and have no problem interpreting various written language formats. They also have no problem converting the output back into speech or properly punctuating the translation. The SMT system is separate from the systems which handle the preprocessing and post-processing. SMT systems operate on a corpus of digital detokenized text. Detokenization means separating the sentence into its tokens such as words and punctuation. SMT begins with this preprocessed corpus in the SL and ends with a body of text in the TL. This output is then post-processed to restore the typical format to the corpus.

2.4 Translation Model

The TM is what allows the SMT system to look up the the joint probabilities of SL-TL pairs. The probability of a given translation is represented by the function ϕ. The TM is much like a bilingual dictionary in that it determines the vocabulary that the system knows. To generate this model, the system uses a parallel corpus of text. A parallel corpus is one that contains a message that has been correctly recorded in both the SL and TL. Larger corpora will provide more statistically significant representations of word correlations and will have a statistically higher probability of containing more words. The quality of the translation is best if the corpus being translated is of a similar domain to the training material [49]. This is not always possible. A problem facing SMT is that only certain languages and subjects have readily available corpora [74]. Parallel corpora are usually generated by human translators which is time consuming and therefore can be costly. Other methods are being explored such as generating the corpora using RBMT systems [42] or automatically generating parallel corpora from unreliable sources like the internet [51] [73]. Commonly, systems will use large parallel corpora from the transcripts of multinational organization like the UN [65] and the European Parliament [47] due to their size and number of language pairs. The technique of creating more useful sub-corpus from larger, less specific corpora has been explored [49].

2.4.1 Word-Based.

Translation models have been built to focus on the different levels of NL organization. The smallest unit of meaning, the word, is a logical place to begin translation. This is arguably the simplest form of SMT, which is known as word-based or lexical translation. This method is like using a digital bilingual dictionary to replace words in the SL with their corresponding words from the TL. IBM Model 1, which was the first of its kind, is a good example of this type of model [23].

The first step in lexical translating is to build the bilingual dictionary. The system will have its own version of a bilingual dictionary that, in addition to storing correlated word pairs, also stores the probability that they are the correct translation. Words often have multiple possible translations into other languages. For example, the German words “Koben”, “Feder”, “Hurde”, or “Stall” may all translate into the English word “pen.” The probabilities of these different translations are represented by $P(t|s)$ where t is the word pen and s is the source German word. Suppose that the probabilities are as follows:

$$\begin{aligned} p(pen|Koben) &= 0.4 \\ p(pen|Hurde) &= 0.3 \\ p(pen|Feder) &= 0.2 \\ p(pen|Stall) &= 0.1 \end{aligned} \tag{2.3}$$

Notice that $\sum_s p(t|s) = 1$ because these four cases represent the full range of translations. Ignoring the calculation of these probabilities for now, one can explore how the system uses them. The probability of a translation of a sentence $S = (s_1, s_2, ..., s_k)$ of length k to a sentence $T = (t_1, t_2, ..., t_m)$ of length m given an alignment function $a : j \rightarrow i$ is determined by Equation (2.4) [23] where t_i is the ith word in T and s_i is the ith word in S.

$$\phi(T, a|S) = \frac{\epsilon}{(l_S + 1)^{l_T}} \prod_{j=1}^{l_T} P(t_j|s_{a_j)} \tag{2.4}$$

ϵ is the normalization constant which is assumed to be a small constant number. This equation calculates the possible translations and their probabilities. Next the generation of $P(t_j|s_{a_j})$ from the parallel corpora is covered.

The parallel corpora consists of a SL corpus and TL corpus that say the same thing. They are sentence aligned so that each sentence is paired with its corresponding sentence in the other language. The translation in this parallel corpora should be extremely accurate because it is how the SMT system will learn the meanings of words. Word alignment is its own field of study [53]. There are subsequently many methods to generate alignments with different claims to their effectiveness. This research will use the method developed by Och which is implemented in the tool GIZA++ [18] and is similar to the IBM Model 1 [58]. Model 1 aligns words using the Expectation Maximization (EM) algorithm [32]. In order to apply EM to Model 1, Brown, et al. [25] created Equation (2.5) to count the number of times two words t and s would be aligned . The funtion δ is the Kronecker delta function given in Equation (2.6).

$$C(t|s;T,S) = \frac{\phi(t|s)}{\phi(t|s_0) + ... + \phi(t|s_{l_s})} \sum_{j=1}^{l_t} \delta(t, t_j) \sum_{i=0}^{l_s} \delta(s, s_i) \tag{2.5}$$

$$\delta(x,y) = \begin{cases} 1 & if x = y \\ 0 & otherwise \end{cases} \tag{2.6}$$

The EM algorithm works by first selecting an initial value for $P(t|s)$. The selection is relatively arbitrary since the EM is guaranteed to converge to a global minimum [23]. The solution is of polynomial complexity so it is not difficult for computers to process. Finally, Equation (2.7) is iteratively applied until the values for $P(t|s)$ converge.

$$P(t|s) = \frac{\sum_{T,S} c(t|s;T,S)}{\sum_t \sum_{(T,S)} c(t|s;T,S)} \tag{2.7}$$

What is left are the word alignments and probabilities $P(t|s)$. This allows us to apply equation Equation (2.4) and translate sentences. The actual generation of translations is known as decoding which it is discussed later.

While Model 1 works, it is not perfect. The translations generated by lexical systems are simply word substitutions that ignore surrounding words and reordering words. Brown, et al. [25] attempted to harness some of these missing pieces of information in subsequent models. Models 2 through 5 are derivatives of model 1. Model 2 builds on Model 1 by adding a factor for reordering words. Reordering is what happens when the index of the words in S are not the same as their correlating words in T. Reordering in Model 2 occurs after the initial lexical translation in Model 1. Model 2 is therefore also of polynomial complexity. Model 3 adds a fertility factor, which allows a single word from S to translate to two words in T. This additional factor makes Model 3 exponential which means the polynomial solutions used in Model 1 and 2 no longer work. To overcome the exponential complexity of Model 3, artificial intelligence (AI) search techniques such as hill climbing to find word alignments are used instead of the exhaustive search in Models 1 and 2. To increase the number of samples found, they use neighbor alignments. Once they have the set of candidate alignments they exhaustively run the EM algorithm on them until the model converges [25]. Model 4 uses relative reordering to make calculating the reordering of large sentences easier. Relative reordering means that words are assigned offsets in the translated sentence instead of an absolute index. It also adds word classes so that the reordering statistics are not based on words but rather word classes. This increases sample sizes of statistic groups. The increased group size contributes more meaningful statistics to train the model. Finally, Model 5 improves on previous models by ignoring possible hypotheses with impossible word alignments. This is known as pruning the search space, and it makes the probabilities of legitimate translation larger and more representative by removing wasted probability mass. For further information about the models, see the foundational paper "The Mathematics of Statistical Machine Translation: Parameter Estimation" [23]. The word alignments and translation probability in these models are the foundations of most modern SMT [58].

2.4.2 Phrase-Based.

Word-based translation is limited because it treats words as the only unit of meaning. A better candidate is the phrase. This is not the linguistic definition of a phrase. It is any contiguous multi-word structure [50]. For example, "over the" is a valid phrase. Phrase based models treat these phrases as the atomic units of meaning, allowing them to interpret contextual meaning more accurately than the word-based models. Where a word-based model would probably suggest the same word for pen in translations of "ballpoint pen" and "pig pen," phrase-based translation will take into account the previous word and make a more accurate translation [80].

The first problem facing this method is the formulation of the phrase translation table. An approach to doing this is to first generate the word alignment from the lexical translation process using the EM algorithm. Phrases are generated from this alignment by adding all contiguous groups of words into the phrase translation table [80]. The phrases much be consistent with the word alignment. For instance, if S_i aligns to two words in T, then a phrase containing S_i must contain both words in T. The phrase translation table contains all possible alignments of S and T phrases. The statistical probability of a given phrase $\bar{s}$ in S translating to a phrase $\bar{t}$ in T is given by Equation (2.8).

$$\phi(\bar{s}|\bar{t}) = \frac{c(\bar{s}, \bar{t})}{\sum_{\bar{s}_i} c(\bar{s}, \bar{t})} \tag{2.8}$$

In this case, c is the count function that counts the number of times a specific phrasal pair is extracted.

Additional features can be extracted from the phrase based TM. The bidirectional translation probabilities can be calculated by reversing $\bar{s}$ and $\bar{t}$ in training and using this as an additional TM feature. Another possible feature is lexical weighting of the components within the phrase. This feature can be generated in many ways and are proposed by a great deal of different research, but its purpose is to augment the phrase-based translation using information from a lexical TM to prevent unlikely phrase-pairs from being weighted too high because the alignment only occurred a few times. Another factor that can be added is a word penalty that effects the score of a translation based on the number of words in it, allowing a system to be tuned for longer or shorter sentences [50].

2.4.3 Tree-Based.

From the communication perspective it makes sense to interpret sentences as strings of words and phrases because it is how they are used. Linguistically however, sentences are broken down into trees, where the root is the full sentence element which is broken down into its phrasal components. These components are broken down further until terminal words are reached. The goal of generating a tree based TM is to generate context-free grammar (CFG) for the SL and TL [39]. To overcome some of the challenges introduced by the tree structure, restrictions are imposed on the model such as forcing it to be a binary tree, disallowing terminals and non-terminals mixing in a grammar rule, and limiting the set of non-terminals. Like the phrase- and word-based TMs, the synchronous context-free grammar (SCFG)s for the tree TM is generated from the parallel corpora. First, phrase-based translation techniques are applied to separate phrases. In the extraction phase, the hierarchical structure of the phrases is extracted as well as the phrase rules. The CFG rules are then generated from the phrase translation list [39].

In this method, syntactic tagging is not required and so the syntactic meanings of the phrases are ignored. The addition of syntactic tags allows for the formation of syntax-aware rules, but this additional constraint causes extra complexity. The grammars generated are usually complex, so the they are often simplified by combining similar rules. The remaining rules are scored by a variety of methods to give preference to the ones most likely to produce a good translation. Additionally, it is common practice to add rules to handle last resort cases not explicitly covered by the model. These rules are called fall-back rules and the metrics must discourage them in order for the TM to be viable. Tree-based models are often equivalent to phrase-based models in their translation quality, and may surpass them when the language pair has extremely different sentence structures [48]. An efficient method exists for decoding these tree LMs, however the topic of tree-based translation is not discussed further because the principles of are not influential in this research.

2.5 Language Model

At this point, the SMT system has the means to statistically correlate the SL and TL corpora. It can also generate a TL translation from a SL corpus. The system has a shortcoming, as it does not have a way of determining and ensuring the fluency the translation it generates. Fluency is how well-formed a translation is based on the standards of the TL.

The LM is the solution to this problem. It is a statistical model of what the TL should look like. It is typically generated from a very large corpus of TL text or, if that is unavailable, from the TL side of the parallel corpus. The system uses the LM to estimate the probability that the sentence T would occur. Ideally this probability would represent the whole of the TL, but it is actually an estimate of the probability that the sentence T would occur in the training data. The LM will rate the fl. Suppose a word based system generates the sentence “mary had a little lamb.” It would equally weight the probability of the output “lamb a had mary little” though this is obviously not proper English. The language model should discriminate against “lamb a had Mary little” and recommend the better translation.

Because the LM represents the probability of a sentence occurring, it would make sense to simply count the number of times a sentence occurs in the training corpora, calculate a probability, and assign this probability to a translated sentence. However, it is unlikely that the available corpus will contain every, or even a small portion, of the sentences the system will need to translate. A majority of the time the LM would score sentences as 0 because the exact sentence was not in the training data. This will be unhelpful to translation and may cause a mistranslation to be unfairly scored simply because it appeared in the training data. Equation (2.9) shows the mathematical representation of this method where T_0 is a sentence of length k made up of words t_i.

$$p(T_0) = P(t_1, t_2, ..., t_k) = p(t_1)p(t_2|t_1)p(t_3|t_1, t_2)...p(t_k|t_1, t_2, ...t_k) \tag{2.9}$$

To prevent underflow, the probabilities are converted into their exponent forms using logarithm base ten. The equation then becomes

$$p(T_0) = p(t_1, t_2, ..., t_k) = exp(log(p(t_1))+log(p(t_2|t_1))+log(p(t_3|t_1, t_2))...+log(p(t_k|t_1, t_2, ...t_k)) \tag{2.10}$$

This method is infeasible for translation because the sentences are unlikely to occur in training. The n-gram is a phrase consisting of n words. It is conceptually the same as Warren Weavers belief that context can be determined by a window of n words. A single word is a 1-gram or monogram. A 2-gram is a set of two words or bigram. When $n = 3$ it is called a trigram and so on. The n-gram is currently the most popular method for developing a LM [48]. Equation Equation (2.9) is an n-gram with $n = k$ where k is the length of the sentence. As n decreases, the chance that the n-gram will occur in the training data increases but the amount of context they contain decreases. Table 2.1 is an example of how a LM may break the sentence “Mary had a little lamb” into n-grams with $n = \{1, 2\}$.

Table 2.1: Monograms and bigrams of the sentence “Mary had a little lamb”.

n=1	<s>	Mary	had	a	little	lamb	</s>
n=2	<s> Mary	Mary had	had a	a little	little lamb	lamb </s>	

The tags “<s>” and “</s>” are extra tokens added to denote the beginning and end of sentences[71]. The value for n can vary. A model usually is built from n-grams with $1 <= n <= x$ where x is some positive integer greater or equal to one. Moses currently has a version compiled to handle values of n up to 12 [70].

With an n-gram, the partial probability is calculated for each phrase of n words. The hope is that the $n - 1$ words preceding word n will give enough information to predict what that n^{th} word should be. To calculate the trigrams for an LM, simply accumulate all trigrams, then use Equation (2.11) to generate the probabilities.

$$P(w_3|w_1, w_1) = \frac{count(w_1, w_2, w_3)}{\sum_w count(w_1, w_2, w)} \tag{2.11}$$

To use the LM to calculate the likelihood of a translation $T = (t_0, t_1, ..., t_k)$, Equation (2.12) is used which is represented in its logarithmic form in Equation (2.13).

$$P(T) = \prod_{i=0}^{k} P(t_i|t_{i-2}, t_{i-1}) \tag{2.12}$$

$$P(T) = exp\left(\sum_{i=0}^{k} log(P(t_i|t_{i-2}, t_{i-1}))\right) \tag{2.13}$$

This model has a flaw, certain n-grams are unseen in the training data and would result in the $P(T) = 0$ for any sentence which contains them. They will cause sentences to get a rating of 0 from the LM regardless of other n-grams. Zero probability would also break the logarithm function. The unseen n-grams problem is remedied by smoothing the LM.

Additionally, as seen in Equation (2.13), the probabilities are reduced using logarithms, and log(0) is undefined.

Smoothing is the method of making probability mass for the unseen n-grams. The most simple approach is to add a constant C to the count of every n-gram, even those which are unseen. While this works, it will cause a great deal of distortion since the probability space is increased by adding C. A better approach is to take some probability mass away from observed n-grams to donate to unobserved n-grams. The result is that no n-grams have zero probability, the sum of probabilities still equals one, and the total count of n-grams is unchanged. A cutting edge implementation of this type of smoothing is Modified Kneser-Ney Smoothing [45]. This method requires two equations. α represents discounted probability of observed n-grams and γ represents the probability of the evidence for a given n-gram occurring. Instead of calculating the amount of the probability discounted as a constant for all n-grams, three values are used that depend on how often the n-grams appear. There are three tiers of discounts, D, based on the number of n-grams [29]. n_1 is the number of n-grams which occur one time, n_2 is the number which occur twice, n_3 is three times and n_4 is four or more. The number of times a particular n-gram $w_1, ..., w_n$ occurs is given by the count function $c(w_1, ...w_n)$.

$$\begin{aligned} Y &= \frac{c_1}{(c_1 + 2 * c_2)} \\ D_1 &= 1 - 2Y\frac{c_2}{c_1} \\ D_2 &= 2 - 3Y\frac{c_3}{c_2} \\ D_{3++} &= 3 - 4Y\frac{c_4}{c_3} \end{aligned} \tag{2.14}$$

N_k is the number of n-grams with a count equal to k. This discount is used in Equations (2.15) and (2.16) to calculate the discounted and subsidized probabilities.

$$\alpha(w_n|w_1, ..., w_n) = \frac{N_{1+}(\bullet w_1, ..., w_n) - D_{c(w1,...,w_{n-1})}}{\sum_w N_{1+}(\bullet w_1, ..., w_{n-1}, w)} \tag{2.15}$$

$$\gamma(w_1, ..., w_n) = \frac{\sum_{i\in\{1,2,3+\}} D_i N_i(\bullet w_1, ..., w_{n-1})}{\sum_{w_n} c(w_1, ..., w_n)} \tag{2.16}$$

Finally, this smoothing is enhanced even more by using the modified Kneser-Ney probabilities in an interpolated backoff equation. This final equation gives us:

$$\alpha_I(w_n|w_1, ..., w_n) = \alpha(w_n|w_1, ..., w_n) + \gamma(w_1, ..., w_n)p_I(w_n|w_2, ..., w_n - 1) \tag{2.17}$$

The result is a smoothed model with intelligently weighted *n*-grams that out performs other smoothing methods including Witten-Bell, Good-Turing, and the constant smoothing [29]. The final formula is shown in Equation (2.18).

$$p_I(w_n|w_1, ..., w_{n-1}) = \begin{cases} \alpha_I(w_n|w_1, ..., w_n) & \text{for } c(w_1, ..., w_n) > 0 \\ \gamma(w_1, ..., w_n) & otherwise \end{cases} \tag{2.18}$$

There are a number of different implementations of LMs with a variety of data structures used to store them. One such implementation is the SRI Language Model (SRILM) originally implemented by Andreas Stolcke [71]. SRILM supports the cutting edge algorithms and methods discussed in this chapter to produce accurate and efficient LMs.

2.6 Decoding

The processes tasked with generating a translation in a modern SMT system is the decoding algorithm. Finding the exact decoding of a sentence was shown to be NP-complete by Knight and Marcu [46]. As with other NP-complete problems, the current approach is to estimate a solution using AI search techniques. An algorithm to do this estimation is beam search.

Decoding approaches translation the way a human does; one segment, which is usually a sentence, at a time. The translator processes one phrase from the source sentence and updates his or her hypothesis as to what the translation is. This is how the decoder works processing one segment at a time and incrementally updating the translation hypothesis as more information from the source sentence is parsed. The algorithm typically decodes the source from left-to-right; however, right-to-left and bidirectional decoding have been researched. They were found to only improved translations in special cases and in certain language pairs such as Japanese and English [76].

2.7 Generating Translations

The TM contains the statistical knowledge of how the SL relates to the TL. Suppose the system must decode the sentence $S_0 = (s_1, ..., s_k)$. The decoding algorithm can take S_0, extract each of the phrases $S_0 = (\bar{s}_1, ...\bar{s}_j)$ that it recognizes in the TL, then order them based on $\phi(\bar{t}|\bar{s})$. This will result in a possible translation, but chances are it would not be accurate. Additionally it would be wasting a great deal of the knowledge encoded in all the other features. So instead, a beam search decoder would start with a null hypothesis $T_0 = \emptyset$ generated from $S_0 = \emptyset$. This is the root of the search tree. The hypothesis is then expanded using the possible translations of the phrases from S_0 added from left to right [50]. It is apparent that the search frontier will expand exponentially and that the search space will be infeasible for even small sentences. There are two primary methods to overcoming this problem.

The preferred method, hypothesis recombination, has no risk of pruning a branch that will lead to the best translation and so it is called error-less [59]. Hypothesis recombination works by combining similar nodes in the search frontier with different scores. These nodes occur when they are generated from different previous nodes. The node with the lower score will be dropped. The effect is a reduced search frontier with the guarantee that only the nodes with the highest potential remain. The search space is still exponential, so another method is required.

The second constraint to the search space is imposed by the beam search. Beam searches limit the search frontier to the best hypothesis as well as those hypotheses which fall within a certain α factor of it, which can be tuned based on whether the system needs to be fast or complete. This approach works to solve the complexity problem, but it inflames a weakness of the heuristics of the search [72].

Beam search is an informed breadth-first search (BFS) algorithm. BFS algorithms check all possible solutions at one level of the search before moving to the next. If someone is trying to figure out a two digit code for a suite case they will most likely use this approach. He or she will start with ‘00’ and trying combinations ‘00’ through ‘09’ then trying ‘10’ through ‘19’ until they find the combination. Changing this one variable is considered a level of the search. Each combination is a node. The set of nodes which are going to be explored are called the frontier. In a BFS, the frontier is stored in a priority queue. During translation, this stack can grow extremely large. A beam search limits the stack with a beam search coefficient $0 < \alpha <= 1$. The beam search will keep a factor within α of best node. In order for this to work, the beam search needs a way to rate the nodes in the frontier. Suppose when testing the suitcase lock, the combinations 03, 04, and 05 were different and indicating that one of these numbers is the correct one’s place. A beam search will keep these three numbers and try them in the next level of search.

The search in translation starts with an empty sentence. The first phrase from S is then added to the source hypothesis and the frontier is populated with all possible translation of this phrase. The nodes are rated using the TM, LM, reordering model (d), word penalty, and any other features which are combined to form a final score. The hypotheses are stored in a priority queue according to this score. Only the best hypotheses in the frontier are kept in a process known as pruning. The size of this queue is determined by an α coefficient which limits the queue to the hypotheses within the α margin of the best hypothesis. At each iteration, another phrase from S is added and every possible resulting translation is added to the search space. The hypotheses are then scored and only those within the α margin of the top of the list are kept. For each node in the frontier, the beam search algorithm uses the next phrase from the S and adds each possible translation to the frontier. The algorithm is complete once all phrases from S are parsed.

The way the individual feature scores are combined is important. Combining them in a log-linear model allows for minimum error training in weighting the features during combination [26] resulting in much better translations. Equation Equation (2.19) shows the log-linear formula where h_i is the value of the i^{th} feature for a given hypothesis in an n-best list and λ_i is the weight of the i^{th} feature.

$$p(x) = exp \sum_{i=1}^{n} \lambda_i h_i(x) \tag{2.19}$$

Values are initialized for each λ_i, and then a bounded optimization of them is performed. For now, assume they are some useable value. This score estimates the quality of the candidate translation.

Thus far the difficulty of translation is not guaranteed to be monotone. This means that at a given level in the search tree, certain hypotheses have dealt with more difficult sections and have a worse score at the time, but may have actually dealt with the section better than other search trees will, resulting in an overall better score. To avoid removing these hypotheses, one can apply the common AI technique of estimating a future cost to discount the nodes which have only dealt with easy translations. This estimation comes from a combination of the features such as the TM based on the certainty of the translation.

Other search algorithms such as A* and greedy hill climbing are used with their heuristics being similar to those used in the beam search [59]. The primary concern is limiting the search frontier so that computational complexity remains feasible without introducing too much search error.

When the search is complete for all segments of the source corpus, the search decoder returns the top hypothesis from each of the searches. Optimization techniques such as Minimum Error Rate Training (MERT) can be applied to help this best translation to be close to the reference translation. These feature optimizations are discussed later.

2.8 Automatic Evaluation Metrics

Evaluating a translation means assigning it a number based on how well it represents the original meaning. The original way to do this was with human translators. Multiple translators were given the original text and the translations. These judges, with their different backgrounds, preferences, and styles would rate the quality of a translation. Their ratings would be combined to form a number which is used to rate the sentence [48]. Though this method is the gold standard for translation quality evaluation, its weaknesses is that it is not automated.

The search for the best automatic evaluation metric is an active area of research [6]. The metric must have a low computational cost so it can be used to quickly rate translations. It should be consistent between runs, meaningful to the quality of a translation, and most importantly it must be similar to a human translation.

One candidate for this metric is bilingual evaluation understudy (BLEU) [60]. Currently one of the most popular metrics, it is an easily computed, corpus level assessment that takes into account both the selection of words and their order. BLEU uses reference translation(s) which must be provided for the algorithm and are typically created by human translators. It generates a score by calculating n-gram precision and relative length between the candidate translation and the references. N-gram precision means the number of n-grams in the reference that are also in the translation. This precision can either be individual, which is the n-gram accuracy for a given value n, or cumulative for values from 1 to n. The calculation can be seen in Equation (2.21). The brevity penalty BP penalizes solutions for being too short and ρ_n is the precision of the n-grams.

$$BLEUn = BP \cdot exp \sum_{i=1}^{n} \lambda_i \log(\rho_i) \tag{2.20}$$

$$BP = \begin{cases} 1 & \text{if } c > r \\ e^{(1-r/c)} & otherwise \end{cases} \tag{2.21}$$

BLEU encourages the use of multiple reference translations. This addresses the important problem of word choice ambiguity. A translation could mean exactly the same thing as the source sentence, but not use the same words as the reference, which would result in a perfectly fluent translation with a very low score. Bringing in multiple translations allows the system to account for some of the variance in language.

Another metric is the National Institute of Standards and Technology (NIST) metric [33]. The NIST metric follows the same basic protocol as BLEU in that it measures *n*-gram precision. The difference is that NIST takes into account the rarity of the *n*-grams, giving more value to *n*-gram matches that occur less frequently in the reference. NIST applies Equation (2.22) to create a weight that is higher for rare *n*-grams and lower for common ones. This approach is based on the theory that uncommon *n*-grams like "low-pressure area" contain more important information than common ones like "in a."

$$Info(w_1...w_n) = log_2\left(\frac{count(w_1...w_{n-1})}{count(w_1...w_n)}\right) \tag{2.22}$$

NIST uses this information with Equation (2.23) to calculate its score. In Equation (2.23) B is the set of all $w_1, ..., w_n$ that co-occur and T is the set of all $w_1, ..., w_n$ that occur in the system output. $\bar{L}_{ref}$ is the average length of the reference and $\bar{L}_{sys}$ is the length of the system output being rated. β is set so the $BP = 0.5$ when $L_{sys} = 2/3 \cdot L_{ref}$. The standard level of NIST is NIST-5 so $N = 5$.

$$NIST = \sum_{n=1}^{N}\left\{\frac{\sum_B Info(w_1...w_n)}{\sum_T(1)}\right\} \cdot exp\left\{\beta \cdot log^2\left[min\left(\frac{L_{sys}}{\bar{L}_{ref}}\right)\right]\right\} \tag{2.23}$$

NIST claims to better represent the score a human rater would provide than BLEU [34]. In this research, both metrics are provided to analyze results; however, the BLEU metrics is most heavily relied on.

2.9 Learning Feature Weights

As mentioned before, the LM, TM, reordering model, and any other features discussed thus far have been combined in a log-linear model. In Equation (2.19), the feature weights

λ_i are initialized to acceptable values such as one. The most popular method for learning the feature weights is currently MERT [56]. MERT establishes a count of errors $E(R_1^S, T_1^S)$ in a translated sentence T given a reference sentence R. This method applies evaluators like BLEU and NIST to tune feature weights. The algorithm then uses this function to obtain a minimum error count $\hat{\lambda}_1^M$ of a translated corpus T by comparing it to a set of K candidate translations $C_s = [t_{s,1}, ...t_{s,K}]$ for a given input s_s using Equation (2.24) with Equation (2.25) defining the value for $\hat{e}$.

$$\hat{\lambda}_1^M = argmin_{\lambda_1^M} \left\{ \sum_{s=1}^{S} \sum_{k=1}^{K} E(r_s, e_{s,k}) \delta(\hat{e}(s_s; \lambda_1^M), e_{s,k} \right\} \tag{2.24}$$

$$\hat{e}(f_s; \lambda_1^M) = argmax_{t \in C_s} \left\{ \sum_{i=1}^{M} \lambda_i h_i(t|s_s) \right\} \tag{2.25}$$

These equations give a rough error count for the different parameters in a log-linear model. In the foundational paper Och, et al. [56] suggested a method of line smoothing which is guaranteed to find an optimal solution. This is done using Equation (2.26) where $t(\cdot)$ and $m(\cdot)$ are constants with respect to γ. This means that they can use the piecewise linear function [61] shown in Equation (2.27) to calculate the intervals in the un-smoothed graph and find the minimum by merging all the interval sets for each of the sentences in the corpus to find an optimal weight.

$$\hat{t}(\hat{s}; \gamma) = argmin_{t \in C} \{t(t, s) + \gamma \cdot m(t, s)\} \tag{2.26}$$

$$f(\gamma; s) = min_{t \in C} \{t(t, s) + \gamma \cdot m(t, s)\} \tag{2.27}$$

Further benefits can be gained by applying boost to MERT [36]. Finally, the methods of tuning these values are deeply coupled with machine learning and many other algorithms can apply such as the Powell search [64] or the simplex algorithm [55].

2.10 Additional Features

SMT systems are not limited to the features discussed thus far. Further features can be added and optimized using the AI techniques discussed. MERT can effectively optimize up to 20 features while other algorithms such as PRO can handle thousands [41]. Some features which have been added including syntactic parsing [79] [75], semantic evidence [20] [52], sentence structure [44], various linguistic features like the number of split contractions in Spanish translations [37], as well as parts of speech and lexical features [40].

In addition to manual selection, features can be automatically selected using automated processes. The standard practice when automating the generation of sparse features is to limit it to the co-occurrence of the most common words [41] [30]. These identified features can be pruned so that only certain ones are selected using a discriminative model to determine which ones are the most informative. The problem that faces all the machine learning approaches is the problem of over-fitting. Over-fitting means that the system is too specialized for the training data, and so the system performed worse than if it were trained less. This can be remedied by stopping the training iterations early [30].

2.11 Augment for Decoding

While the decoding is a strong focus for feature optimization it is not the only one. The is are drawbacks to modifying this algorithm. Adding complex features to decoding will slow the algorithm down. Recall that the feature must be calculated for every partial hypothesis. The system also has limited knowledge of the problem during decoding. The translation evidence and features are nearly always based on the segment that is being translated. Finally, it can be time consuming to modify the decoding algorithm and modifications will have little, or negative, impact on the translation. n-best list re-ranking is an augmentation for decoding which alleviates the aforementioned problems and can be used instead of modifying the decoding algorithm [57]. It allows new feature sets to be tested in a more expedient manner. Additionally, a re-ranking problem is much easier to solve than the beam search used during decoding. The tradeoff is that the improvements of *n*-best list re-ranking cannot be better than the best hypothesis already in the *n*-best list . Therefore, re-ranking is not a substitute for a quality decoding algorithm; however, it is a useful augmentation that can produce statistically significant improvements[57].

The systems that re-rank *n*-best lists after decoding are known as two-pass systems. They apply the additional or modified features with the purpose of re-ordering the *n*-best list to achieve a better translation [28]. The first-pass is the decoding process in which the *n*-best lists are created. During the second pass the new features are combined with old ones to re-rank the *n*-best lists. Usually the metrics are applied in the same way as during the decoding process; however, some researchers have explored new ways to combine the second pass scores. For example, Sokolov, et al. [69] proposed a method of applying a non-linear score function to the post decoding re-ranking. It is this extra information and freedom of complexity which *n*-best list re-ranking usually attempts to harness [20] [66] [17].

III. Design and Methodology

This research proposes an *n*-gram language model (LM) consisting of monograms, bigrams, and trigrams extracted from the *n*-best lists generated during decoding. The model is then used to re-rank the *n*-best lists to improve the quality of the translation produced by the system. Figure 3.1 shows where the re-ranking takes place within the statistical machine translation (SMT) process. The re-ranking process is broken down further in Figure 3.2.

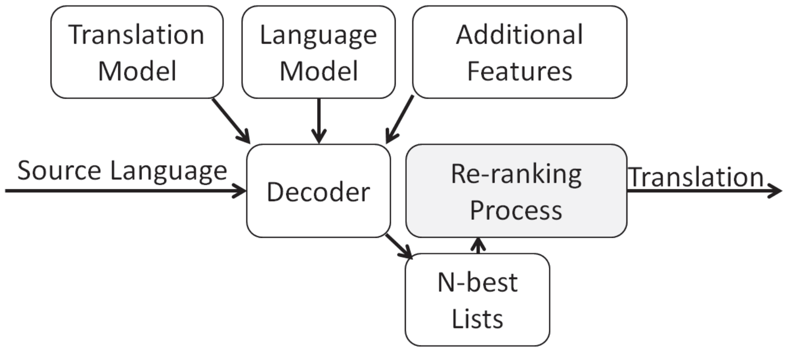


Figure 3.1: Location of the Re-Ranking Process within the SMT process.

The input to the process is the *n*-best lists generated during translation. There is one *n*-best list for each segment, which is usually a sentence, in the corpus. Each list consists of *n* hypotheses. The process involves generating the *n*-gram LM from the *n*-best lists which is referred to as model generation. It also involved applying the LM to the original *n*-best lists to re-rank them.

The count function is part of the model generation steps. It counts the *n*-grams that occur in a single *n*-best list. In the process, this individual count is made for every *n*-best list in the corpus. Because the *n*-best lists are not a trusted source of language, the counting method incorporates a weighting parameter. When counting the *n*-grams in an *n*-best list, this parameter is the value that is counted for each occurrence. The final count is then rounded up to the nearest integer. For example, if the weighting parameter for a hypothesis is 0.4 and the *n*-gram "one two three" appears in it twice, the overall count of "one two three" occurrences within the *n*-best list will only increase by one. The weighting parameter allows the count function to trust certain hypotheses more than others.

The combining function is also part of the model generation steps. It combines the *n*-gram counts for individual *n*-best lists into an aggregate count. This function enables the re-ranking process to combine *n*-gram counts over a certain scope. In addition, the combining function takes a weighting argument during one method of operation which allows different *n*-best lists to be treated with different degrees of trust.

The final step in the model generation part of the re-ranking process involves using SRILM to convert the *n*-gram counts into a LM. This is done using a default compilation of SRILM. The conversion not only calculates the probabilities of each *n*-gram, but it also applies modified Kneser-Ney smoothing with back-off and interpolation.

The final operation of the re-ranking process is to score the hypotheses by the newly generated LM and sort the lists according to their new scores. The values in the LM are $p_I(w_n|w_1, ..., w_{n-1})$ for the monograms, bigrams, and trigrams in a hypotheses. These probabilities are used with Equation (2.13) to rate each hypothesis in the *n*-best list. The new score for the hypothesis calculated using Equation (3.1) where x is the hypothesis, $P'(x)$ represents the new score, $P(x)$ is the old score, λ is the feature weight, and h is the score assigned to the hypothesis by the LM. The *n*-best lists are then ranked according to the new score of their hypotheses.

$$P'(x) = P(x) + \lambda' h' \tag{3.1}$$

The new feature is weighted according to a feature weight that is learned in a training process discussed later. Every *n*-best list in the corpus is then ranked according to this new score. The top hypothesis of each *n*-best list are combined as the new translation.

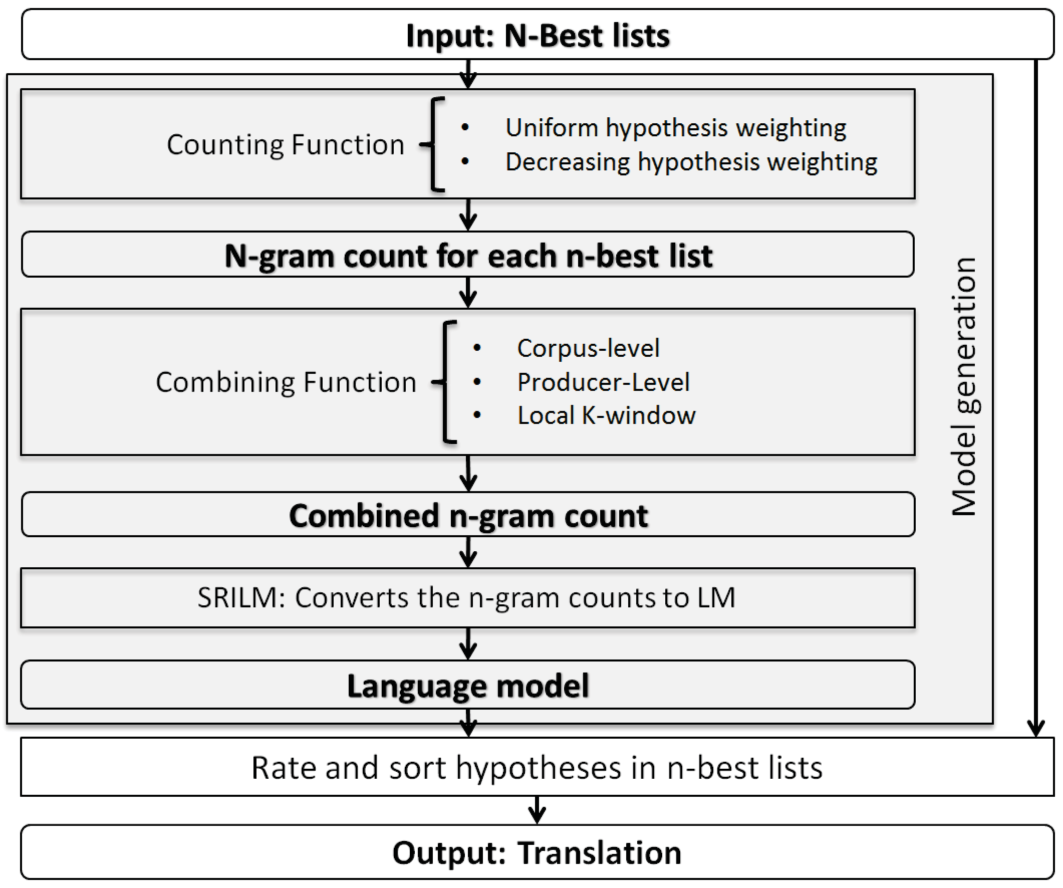

Input: N-Best lists
Counting Function
Uniform hypothesis weighting
Decreasing hypothesis weighting
N-gram count for each n-best list
Combining Function
Corpus-level
Producer-Level
Local K-window
Combined n-gram count
SRILM: Converts the n-gram counts to LM
Language model
Model generation
Rate and sort hypotheses in n-best lists
Output: Translation

Figure 3.2: The Re-Ranking Process.

There are three primary scopes for which the LMs are made: corpus-level, producer-level, and local *k*-window. For each scope, two weighting schemes are used with the counting function to generate the *n*-gram counts. The first is a uniform weighting where each hypothesis is given a weight of one. This is the default and it treats all hypotheses within an *n*-best lists equally such that one occurrence of an *n*-gram is counted as one. The second weighting scheme applies a weight which starts at 1 and degrades to 0.1 for the last hypothesis in the *n*-best list. The degrading value is meant to leverage the possibility that the better hypotheses will be higher in the *n*-best list.

Prior to counting *n*-grams in the *n*-best lists, the duplicate hypotheses must be removed. The occurrence of duplicate hypotheses is due to multiple search paths leading to the same phrase. The number of occurrences of a single hypothesis in a *n*-best list is most likely caused by the phrases it is build from and not due to the quality of the translation. Additionally, some *n*-best lists may only have one or two unique hypotheses. Keep-best recombination is applied to all *n*-best lists before they are processed to remove the duplicates without affecting the translation score. Analysis of recombination and varying methods to apply it are discussed later.

3.1 Corpus-Level

The corpus-level approach generates a LM for the entire corpus. This means that all *n*-best lists in the corpus are used to generate a single LM that is applied to re-rank all *n*-best lists in the same corpus. This scope is selected based on the hypothesis that there will be similar enough word usage within a corpus that the LM will rate hypotheses higher which better conform to the leading *n*-gram usage. These translation hypotheses which conform the best should be the better translations and the resulting corpus-level translation should have a higher BLEU score. For example, a corpus of all news stories should contain the common theme of terminology generally used by reporters, and hypotheses which contain these common terms are likely to be more correct. Figure 3.3 is a graphical representation of the process.

Building a LM from *n*-best lists is different than building one from the training corpora. The training corpora are trusted to be fluent while the *n*-best lists generated by the translation system are not trusted. To avoid learning poor fluency from the *n*-best lists , the small values $n = \{1, 2, 3\}$ are used. As mentioned in Chapter 2, small values of n check for adequacy while larger values check for context. Small values also increases the chance that the *n*-grams will co-occur. The *n*-best list also represents language in non-standard way so the standard LM tools will not work to generate the required models which is why the counting function and combining function are required.

The counting function is applied to each *n*-best list in the corpus. Two *n*-gram counts are generated, one with uniform weights and one with declining weights. The *n*-gram counts for all *n*-best lists in the corpus are processed by the combination function which combines all the *n*-gram counts with a uniform weight of one. The combined *n*-gram count is process by SRILM's "ngram" function with the "interpolate" and "kndiscount" flags set to convert the *n*-gram count into a LM. This is done to maximize speed since SRILM is optimized and written in a compiled programming language. It also reduces repeated work since a great deal of research and effort have gone into making SRILM efficient and reliable. The result is an ARPA formatted [1] *n*-gram LM with smoothing.

The process of generating the LM is done twice. Once to create the LM using the uniform weight method of the counting function, and once to create a LM using the declining weights. Both LMs are applied by the re-ranking function the same way.

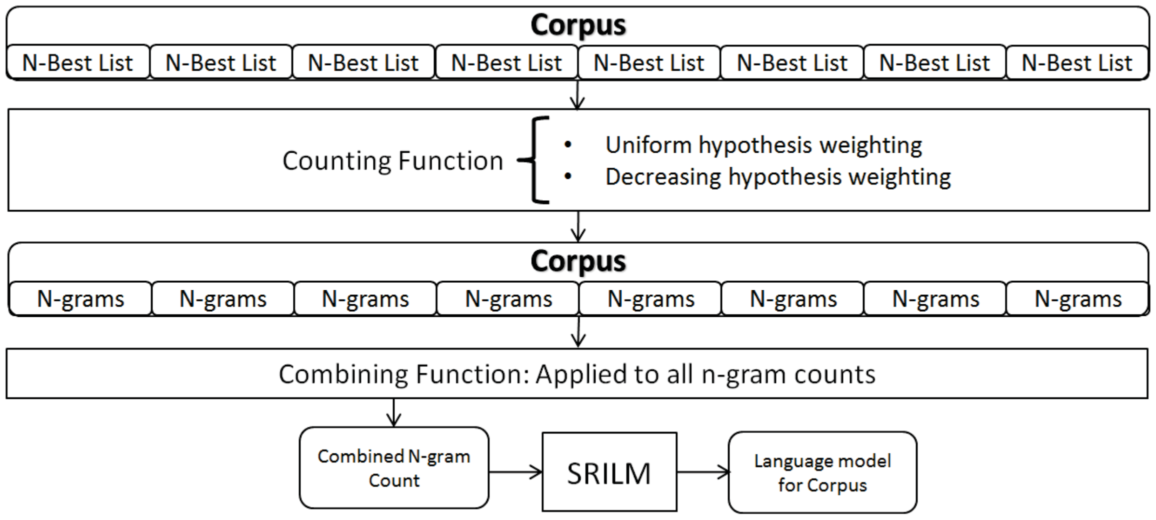


Figure 3.3: Data flow during corpus-level LM generation.

The feature weight for the re-ranking process is calculated by doing a search for feature weights between -1 and 1. This is done by giving the feature an initial weight of -1 and incremented in 200 steps to the value of 1. At each iteration, the re-ranking function is applied using the LM generated with uniform weights. The BLEU score of the translation is stored with the feature weight which produced it so that the maximizing weight can be selected. This method is used instead of MERT, PRO, or any other optimization algorithm because the values will be used to form a graph of how the feature affects the BLEU score. This is a feasible because only one feature is being adjusted and the feature dominated the order of the *n*-best lists as the weight approaches the extrema. An additional test is performed for a larger range of values to ensure that the optimal feature weight does not exist outside of the -1 to 1 bounds. This learning process is applied to the *n*-best lists of TST11AR, TST10AR, and DEV10AR. If the maximizing weights are similar it indicates that the feature is viable for use in translating unknown translations. The weight which maximizes the translation of TST11AR will be used to translate TST10AR and DEV10AR.

3.2 Producer-Level

A more granular distinction than the full corpus can be made when separating word choice tendencies. Isolating the different producers in the corpus will allow the model to represent a person's particular style of communication. The hypothesis is that individual speakers will have a tendency to focus on a particular topic and will have a relatively homogenous word choice. The datasets are from Technology, Entertainment, Design (TED) talks, and therefore the producer is the speaker delivering the talk with the occasional interjection from the host. Identifying separate producers in a corpus based on the content is another topic of research. Sometimes the distinctions are provided within the medium and simply need to be identified and recorded for them to be of further use. For example, in a newspaper the articles each typically written by a single author and often have the author's name printed with them. If the system is being trained to translate a digital news source from another country, then it would build author specific language models based author. The datasets for this research are TED talks and were provided with the separate talks annotated by Extensible Markup Language (XML) tags. Using these tags, the first sentence of each talk within the corpus is identified and its sentence ID is recorded. These IDs are stored in a list. The TST11AR set contains 16 talks, TST10AR contains 13, and DEV10AR contains 9. Figure 3.4 is a graphical representation of generating the producer-level LM.

First generate an *n*-gram count for each *n*-best list using the counting function. The combining function then combines the *n*-grams per producer so each speaker will have a combined *n*-gram count. These combined counts are then individually converted into LMs for each producer using SRILM. This process is repeated twice, once with a uniform weight in the counting function and once with the decreasing weight in the counting function to produce two LMs.

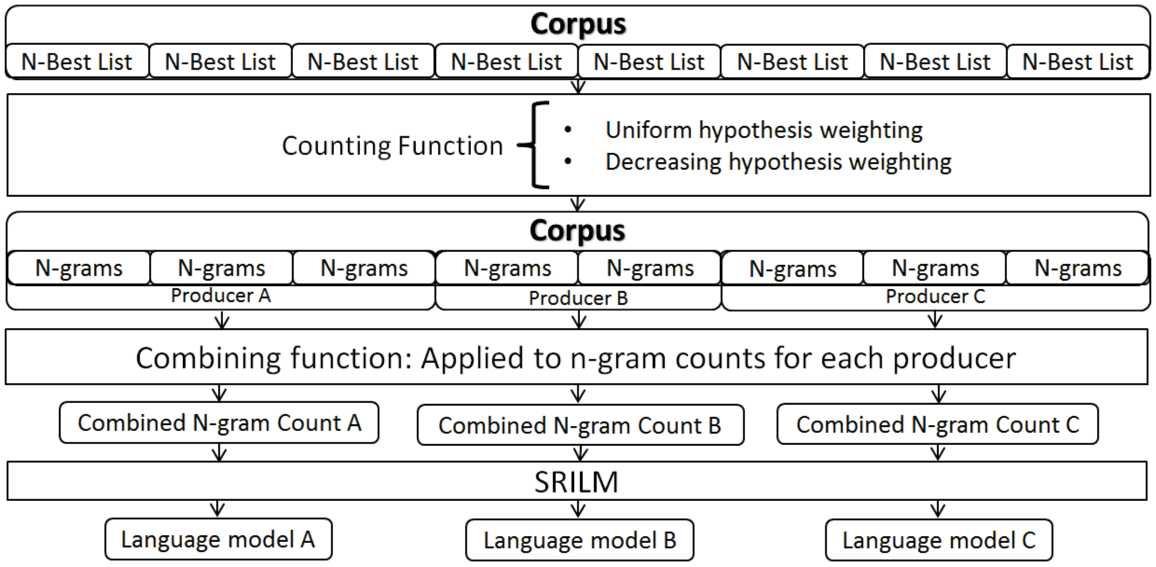


Figure 3.4: Data flow during producer-level LM generation.

The *n*-best lists for a producer are rated by the LM built from their *n*-gram counts. The LM is applied to each *n*-best list so that each hypothesis has a rating. The feature weight for the LMs can be learned once for all producers' LMs even though each individual is being rated against a different model. What is actually being rated is how well producers adhere to a homogenous style and whether or not that adherence is enough to aid translation. The feature weight is set to the same feature weight that was used in the corpus-level.

3.3 Local K-Window

The smallest scope is the local k-window. This method uses the n-gram counts from a window of k surrounding n-best lists to form the LM that is used to re-rank the center sentence. The experiments will be conducted using values of $k = 7$ and $k = 11$. This means that the models will not be lexically rich. They will not be able to represent overarching themes well. Instead, the theory inspiring the local k-window LM is similar to the basis of n-grams. It is an attempt estimate the probability of a sentence given the surrounding sentences. This is done by creating a LM from the n-gram occurrences in the neighboring n-best lists then re-ranking the focus n-best list based on this LM.

Implementing the local k-window LM begins with the same counting function. The function generates n-gram counts for each n-best list in the corpus. The combining function is given the corpus of n-gram counts as well as a weighting array. The weighting array consists of k real numbers. Equation (3.2) is used to select which of the n-gram counts to use when combining them for the focus sentence i using a window of k. The numbers in the weight array are multiplied by the counts in their corresponding n-gram count. These counts are rounded up to the nearest integer and combined to form the combined n-gram count for the focus sentence.

$$L = \begin{cases} H_0, ..., H_k & \text{if } \frac{k}{2} > i \\ H_{|N|-k}, ..., H_{|N|} & \text{if } i + \frac{k}{2} > |N| \\ N_{i-\frac{k}{2}}, ..., H_{i+\frac{k}{2}} & \textit{otherwise} \end{cases} \tag{3.2}$$

A combined n-gram count is generated for every n-best list in the corpus. Figure 3.5 illustrates how the k-window for a single focus sentence i may be generated while Figure 3.6 illustrates the overall process of generating local k-window LMs.

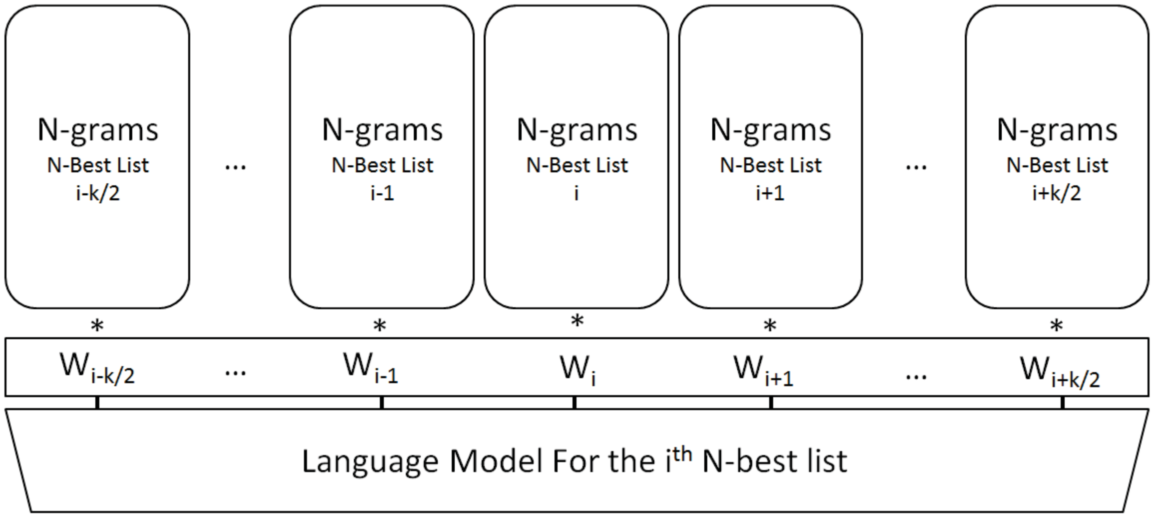


Figure 3.5: Combination with feature weights in the local model.

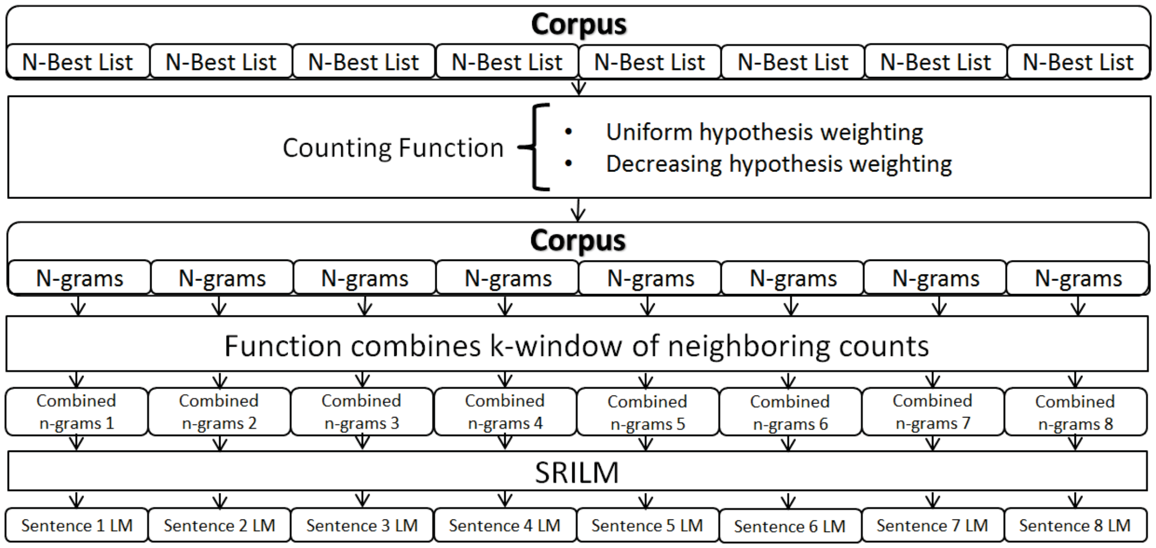


Figure 3.6: Data flow during producer-level LM generation.

The proximity of an n-best list to the focus n-best list may change the value of co-occurring n-grams. To account for this, various weighting arrays for combining the local n-gram counts are tested. The first method is the uniform combination of sentences with the window $k = 7$. This is done by passing the array $\{1, 1, 1, 1, 1, 1, 1\}$ to the combination function. There is a chance that including the focus sentence in the model is detrimental to the translation. To test the effect of removing the focus sentence, the weights of $\{1, 1, 1, 0, 1, 1, 1\}$ are tested. Another possibility is that the n-gram repetition will peak at a certain distance from the focus hypothesis. The opposite may also be true, and so the weights of $\{2, 1.75, 1.25, 1, 1.25, 1.75, 2\}$ and $\{1, 1.25, 1.75, 2, 1.75, 1.25, 1\}$ are tested. In addition, uniform weights with $k = 11$ is tested to see if there is an improvement. If increasing k by the small margin improves the performance of the uniform weights, the same should hold for other weighting schemes.

The process of generating local k-window LMs are repeated for each weighting scheme and for both weighting methods of the counting function. When a corpus is re-ranked, each individual n-best list is re-ranked using Equation (3.1) with the score from its corresponding LM. The feature weight from the corpus-level model is used with this features.

3.4 Software

The experiments are conducted on three workstations. A Windows workstation with Microsoft Windows 8.1 64-bit [5], an Ubuntu Desktop 12.04 64-bit [13] with Linux kernel 3.5, and a Scientific Linux version 6.0 [11] with Linux kernel 2.6. The Scientific Linux workstation is supported by a Sun Grid Engine (SGE) [7] cluster which was used to generate the Arabic-English datasets n-best lists. The primary SMT software used for decoding in this research is the open source Moses distribution version 1.0 [70]. The preprocessing and post-processing of the data is accomplished using the Perl scripts provided with the Moses distribution. The scripts are referenced by their names, less the ".perl" or ".pl" extension, when used. They were originally developed by various authors as is normally the case with open source projects. Moses was compiled using the GNU Compiler Collection [2] compiler on both a Linux 2.6 kernel and a Linux 3.5 kernel. It was compiled with support for SRILM [71] .

The LMs are generated using SRILM toolkit version 1.6 which is downloaded as source code and compiled on Windows 8.1 using Microsoft Visual Studio 2013 Ultimate. A majority of research experiments are conducted using Python [9] version 2.7 with interpreters on Windows 8.1 64-bit, Linux 2.6 and Linux 3.5. ActivePerl version 5.16.3 for 64-bit Microsoft Windows with multithread support is used for Perl scripts on the Windows workstation. The Perl interpreter from the Debian repositories is used to interpret Perl scripts on the Linux workstations.

3.5 Data

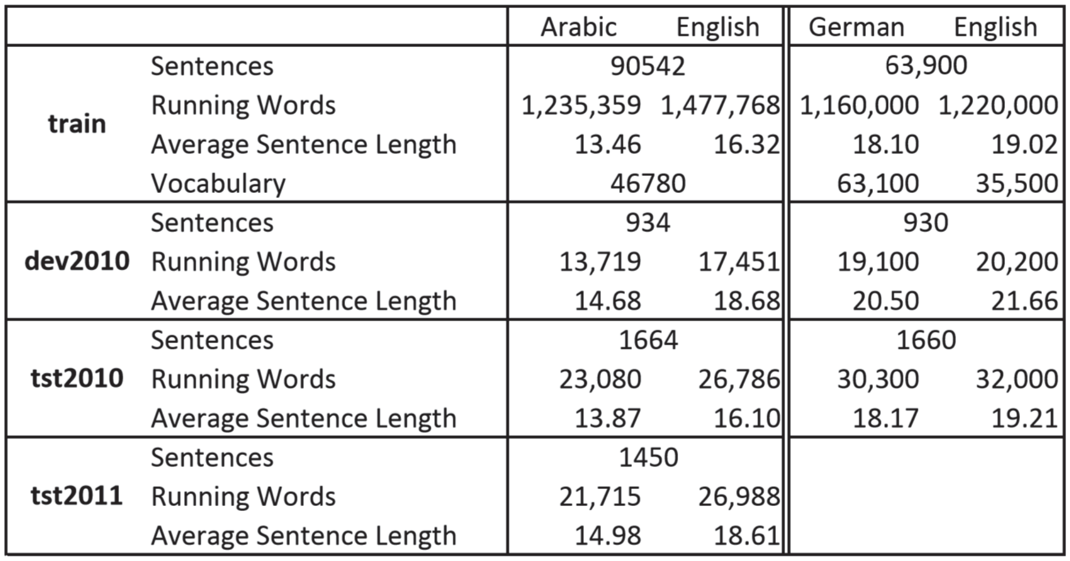
There are three primary datasets used in the research. These are the Arabic-English 2011 evaluation campaign text, the Arabic-English 2010 test corpus, and the Arabic-English 2010 development corpus from the 2011 International Workshop on Spoken Language Translation (IWSLT), which will be referred as TST11AR, TST10AR, and DEV10AR respectively. The *n*-best lists created from these datasets are the primary focus of the research. The *n*-best lists for the Arabic-Englishdatasets were created using a cutting edge SMT system implemented by members of the Department of Defense (DoD), Air Force Research Laboratory (AFRL) Human Effectiveness Directorate, and the Massachusetts Institute of Technology Lincoln Laboratory (MITLL) Human Language Technology Group for the IWSLT 2011 evaluation. The system employs a number of the methods discussed in Chapter 2. The *n*-best lists from TST11AR, TST10AR, and DEV10AR were generated when the system was translating for the IWSLT evaluation. The system is documented in a paper [19] published by the DoD-MITLL-AFRL group detailing their methodology and results.

The second datasets are the German-English IWSLT 2010 development and test corpora referred to as TST10DE and DEV10DE respectively. These datasets were translated with a standard Moses version 1.0 system compiled on the Ubuntu workstation. The system is trained on the German-English 2010 training data. Both the Arabic-

English and German-English datasets are available for download from Web Inventory of Transcribed and Translated Talks (WIT^3) [27]. All IWSLT sets contain the source language for the TED talks. They also contain the parallel reference translation. In addition, the datasets include training parallel corpora for its language pair and a large monolingual corpus in the TL for LM training.

The details about the datasets are listed in Table 3.1. Prior to generating these numbers the datasets are lowercased and tokenized. This is done using Perl scripts from the Moses suite.

Table 3.1: Dataset statistics.



		Arabic	English	German	English
train	Sentences	90542		63,900	
	Running Words	1,235,359	1,477,768	1,160,000	1,220,000
	Average Sentence Length	13.46	16.32	18.10	19.02
	Vocabulary	46780		63,100	35,500
dev2010	Sentences	934		930	
	Running Words	13,719	17,451	19,100	20,200
	Average Sentence Length	14.68	18.68	20.50	21.66
tst2010	Sentences	1664		1660	
	Running Words	23,080	26,786	30,300	32,000
	Average Sentence Length	13.87	16.10	18.17	19.21
tst2011	Sentences	1450			
	Running Words	21,715	26,988		
	Average Sentence Length	14.98	18.61		

3.6 Data Processing

The experiments performed in this research are scripted to be easily repeatable. The primary programming language used in the research is Python. This is the language the experiments are implemented in and also the language used to drive them. In addition, Bourne-Again Shell (Bash) scripts were used for Linux tasks which simply require command line commands to be executed. To facilitate dealing with *n*-best lists , a data structure is created to encase the necessary information for each *n*-best list. In Python, these data objects are known as classes. The Python array object is a list. The *n*-best list object contains a list of the *n* translation hypotheses created for the *n*-best list. Each entry is a list which contains the information from one line of the *n*-best list file split up into each individual element. The class also maintains the reference and source sentences as well as the file names that contain them. The class allows *n*-best lists to be manipulated like objects in code facilitating easier research and streamlining hypothesis verification. The class also implements functions to modify and retrieve the *n*-best list's data.

3.7 Initial Decoding

The primary datasets are TST11AR, TST10AR, and DEV10AR. Again, the steps taken and methodology followed to produce their corresponding *n*-best lists are outlined in the IWSLT paper [19]. The *n*-best lists contain a set of nine feature values for hypotheses that are unweighted and a score that is weighted. To verify that the experimental functions handle data correctly, re-ranking is tested against using the features and weights from decoding. These weights were generated iteratively through MERT. Table 3.2 outlines the training parameters and final weights from the Arabic-English *n*-best list generation while Table 3.3 defines what each feature is.

Table 3.2: Feature weights and bounds.

	TM_1	TM_2	TM_3	TM_4	TM_5	TM_6	LM	w	d
max	1.5	1.5	1.5	1.5	1.5	-2.0	1.5	1	1.5
min	0	0	0	0	0	-2.0	0	-1	0
init	1.0	0	0.25	0.25	0	-0.2	1	0	0.01
MERT	0.9564	0.1275	0.3808	0.0017	0.2582	0.0	0.4568	-0.9647	0.6648

Table 3.3: Feature definitions.

TM_1	P(s|t)	TM_4	LexW(e|f)	d	Distortion
TM_2	P(t|s)	TM_5	Phrase Penalty	LM	6-gram language model
TM_3	LexW(f|e)	TM_6	Lexical Backoff	w	Word Distortion

The decoder uses a beam search algorithm. The α parameter for the beam search is set to 0.01. The decoder is also set to generate an *n*-best list with 200 hypotheses in it. The *n*-best lists are non-distinct which means that the *n*-best list may contain repeated hypotheses with different weights. The Moses task is configured to run parallel on 32 SGE nodes with at least 40 Gb free memory. The output, intermediate files, and configuration files are separated due to the parallelization and are aggregated into single files for *n*-best list , source, and reference for each dataset.

As outlined in the paper, the *n*-best lists produced by Moses are further optimized to generate the final selection. In this research, the aim is to generate a method that can be executed directly after Moses decoding. Therefore, the *n*-best lists output by the Moses system with the feature weights outlined by the configuration file are used as the baseline with the DOD-MITLL-AFRL system as the goal which defines ultimate success.

3.8 Scoring Baseline and Criteria

These translation quality ratings are generated using the mteval_v11b Perl script distributed with Moses and used during the IWSLT evaluation. This script calculates the individual and cumulative score for BLEU-*n* and NIST-*n* evaluation metrics for values of *n* from 1 to 9. The final score reported by the script and the one used for the IWSLT are the cumulative BLEU-4 and the cumulative NIST-5 scores. These are the scores which are used as the primary measure of translation quality in this research. The scores from the evaluation script used on the AFRL system output are used as baselines for that translation.

The script is also run on the output of the German-English system to determine a baseline score.

The possible improvement in *n*-best list re-ranking is limited by the best hypotheses translations in the *n*-best lists [57]. The BLEU-4 metric is used to programmatically calculate the quality of the candidate translation. This is done through an oracle re-ranking function. The oracle function calculates the estimated BLEU-4 score for each hypothesis in the *n*-best lists. The sentence-by-sentence BLEU estimation allows us to select a hypothesis which should maximize the final document score [57]. The BLEU-4 estimate of each hypothesis is used as its score and the *n*-best lists are re-ranked. The new top hypothesis of each *n*-best list is then used as the candidate translation. The BLEU score of the corpus is the upper bound of achievable BLEU score via re-ranking the *n*-best lists. Additionally, the re-ranking based on the score can be reversed so that the hypotheses are sorted lowest to highest. The translation's BLEU score will now represent the lowest achievable score.

3.9 Displacement

A metric is defined that represents how much a list is re-ranked from its original ordering. This metric is referred to as displacement. It measures the effect a re-ranking has on the order of an *n*-best list. A method is implemented to quickly calculate this metric. The metric is used to debug re-ranking tests and aid in interpreting results of re-ranking. If displacement is high then the metric caused a great deal of changing in the lists where as a displacement of zero means the list is unchanged.

Simply put, displacement is the number of swaps required to cause the changes in the list between the origin and the current arrangement. This is calculated using a modified merge sort algorithm that returns the number of swaps used to sort the list instead of the sorted list. Merge sort has a complexity of $O(n\ log(n))$ and will easily process any *n*-best list created by decoding [21].

The sorted list is not returned because it will be $\{0, 1, ..., n\}$ every time. As an example, let a 6-best list be represented by $H = \{h_0, h_1, h_2, h_3, h_4, h_5\}$. Its displacement is 0. Suppose the same list is re-ranked by a feature causing it to be arranged as $H = \{h_1, h_0, h_2, h_3, h_4, h_5\}$. Its displacement is now one because it would take one swap to return the list to order. The maximum achievable displacement for a given value of n is equal to $\sum_{i=1}^{n} i = \frac{n^2-n}{2}$. The displacement count can be normalized by dividing it by the maximum displacement to yield a displacement percentage. This helps give the displacement perspective and is also useful when comparing n-best lists of varying sizes. The function used in the research is implemented in Python. This measure for displacement is useful because it counts swaps and the score for a system can only change when hypotheses in the n-best lists swap.

3.10 Recombination

Recall from Chapter 2 that recombination is used during decoding to limit size of the search space. It is done by combining matching nodes in the search tree and keeping the highest scoring hypothesis. When generating the n-best list , recombination is not applied. This is because there will be no further expansion of the search space and so reducing the frontier is not necessary. Second, it is likely that a great deal of the hypotheses are repeated and it takes extra time to search through the different hypotheses in order to find n unique ones. In the case of small sentences such as “thank you” there may not even be n unique hypotheses.

There are three possible improvements that can be made in light of this practice. First, by performing recombination before generating the n-best list, the chance of including a better hypothesis is increased within the list is increased. To test this hypothesis, a Moses SMT system is trained using the train-moses Perl script included with Moses. For training, the training corpora from the 2010 German-English datasets are used. Only one round of training is performed and the very first system generated by the train-moses script is used. The script handles the preprocessing of the data such as tokenizing and down-casing. The

LM for the system is generated by SRILM. The English corpus is tokenized and downcased first. The LM is generated using modified Kneser-Ney smoothing with backoff and interpolation. All other options are left at their default settings. The TM is generated automatically by the Perl script. It uses the parallel train.de-en.de and train.de-en.en corpus. Default phrase reordering smoothing is used.

Translations of TST10DE and DEV10DE are created using the newly trained German-Englishsystem. *N*-best lists of size 200 are created. In addition to standard *n*-best lists, distinct *n*-best lists are also created using the $-distinct$ option with moses. This option essentially performs recombination before generating the *n*-best list in an attempt to generate *n* unique hypotheses. These two types of translations for the two datasets are repeated ten times and the execution times are recorded for each of the 40 distinct translations. The hypothesis is that the distinct *n*-best lists will have a higher achievable re-ranking score but may require more time to generate.

In some cases, generating distinct *n*-best lists may not be an option. The timing experiment will determine the extent to which generating distinct *n*-best lists increases the time requirements of decoding. In the cases where timing is important, an alternative method can be applied. Instead of generating an *n*-best list of unique hypotheses of size *n* to improve accuracy, the second experiment will perform recombination on the normal *n*-best list to reduce the problem space. To analyze the possible speed improvements, each *n*-best list file from the German-English and Arabic-English datasets are parsed via a simple Python script which reports the length of the *n*-best list before and after recombination. Recombination is achieved by iterating through the *n*-best list , caching each hypothesis as it is observed. If a hypothesis is already in the cache, it is removed from the list. A full corpus is parsed and stored as a list of these *n*-best list objects. The recombination function is applied iteratively on each element of the list. The results of the recombination are recorded in a file for analysis.

The final experiment involving recombining n-best lists varies the recombination technique. Generally the keep-best method is used. As mentioned in Chapter 2, this method keeps the hypothesis with the highest score when two hypotheses match. This experiment tests if recombining hypotheses in other ways will improve a score. This may be the case if the density or location of duplicate hypotheses in the n-best list correlates to the quality of the hypothesis. The past two experiments used the standard keep-best method. This will not affect the score of the translation because it will never remove the top translation from the list. Keep-worst tie-breaking will likely result in a worse translation. To verify this, keep-worst tie-breaking of the list is kept using the same method as keep best, except that the list iterates from the last to the first element.

The final method of recombination is achieved through combining feature values of the individual hypothesis using a linear equation. For each of the features, the scores of all matching hypotheses are combined using h_i to represent the i^{th} hypothesis and λ_i to represent the weight given to that hypothesis in the final combination. One method of the experiment uses uniform values of λ to average the scores of matching hypotheses when as using the formula $\lambda_i = 1 - \frac{0.9}{n} \cdot i$ so if the first hypothesis in the list is combined with the last, the first is weighted with the value 1 and the last hypothesis is weighted with the value of 0.1 when they are combined. These new feature values are combined using the standard log-linear equation with the final system weights found in Table 3.2 to generate new final scores for the hypotheses. The new scores are used to re-rank the n-best lists and generate a hypothesis translation which is then evaluated using the mteval_v11b script.

IV. Results and Evaluation

4.1 Establishing a Baseline

The bilingual evaluation understudy (BLEU) and National Institute of Standards and Technology (NIST) scores are both used to rate the quality of translations with the emphasis on BLEU. There are two baselines for the datasets TST11AR, TST10AR, and DEV10AR. One is the score of the original ranked *n*-best lists from the 10^{th} iteration of the Moses's MERT training performed by the DoD-MITLL-AFRL system [19]. This baseline is refereed to as NB and it is used to show the improvement of translation made by the re-ranking process. The second baseline is the score of the output generated by the MITLL-AFRL-DOD system. This output was generated after further optimization and will be used as a reference for whether the improvements made by the re-ranking process are significant. The baseline is referred to as MA. The baseline scores for TST10DE and DEV10DE are the BLEU scores of the translation using the system. The $-distinct$ option does not effect the translation. It only effects the elements in the *n*-best list but not the top hypothesis. These scores are shown in Table 4.1. The bilingual evaluation understudy (BLEU) and National Institute of Standards and Technology (NIST) scores are both used to rate the quality of translations with the emphasis on BLEU. There are two baselines for the datasets TST11AR, TST10AR, and DEV10AR. One is the score of the original ranked *n*-best lists from the 10^{th} iteration of the Moses's MERT training performed by the DoD-MITLL-AFRL system [19]. This baseline is refereed to as NB and it is used to show the improvement of translation made by the re-ranking process. The second baseline is the score of the output generated by the MITLL-AFRL-DOD system. This output was generated after further optimization and will be used as a reference for whether the improvements made by the re-ranking process are significant. The baseline is referred to as MA. The baseline scores for TST10DE and DEV10DE are the BLEU scores of the translation using the system. The

$-$*distinct* option does not effect the translation. It only effects the elements in the *n*-best list but not the top hypothesis. These scores are shown in Table 4.1.

Table 4.1: Baseline translation scores.

	TST11AR		TST10AR		DEV10AR		TST10DE	DEV10DE
	NB	MA	NB	MA	NB	MA		
BLEU	0.2195	0.2350	0.2146	0.2477	0.2337	0.2478	0.2005	0.2110
NIST	5.5152	5.7555	5.7684	6.0184	5.8201	6.0199	6.0411	6.1797

Note that the score for the German-English systems translation of TST10DE and DEV10DE are not meant to be competitive. The system is meant to be an easily implemented baseline system as opposed to the highly optimization system used for the Arabic-English translations. The German-English system is used to test the change in achievable score between distinct and non-distinct *n*-best list generation. The baseline for the Arabic-English system is meant to represent the baseline for translation improvement. The goal of the research is to increase this score. Note that the NIST and BLEU metrics are not linearly related. The BLEU score of DEV10DE is less than the score of DEV10AR while the NIST score is higher. This is because of the extra emphasis NIST places on rare *n*-grams. There are merits to both evaluation metrics and this research will attempt to improve both.

4.2 Limits of Improvement

Upper and lower limits of improvements are estimated using BLEU oracle re-ranking. Table 4.2 shows the estimated upper and lower limits of the achievable score by re-ranking. The results indicate that TST11AR can achieve an improvement of 22.81%, TST10AR can improve by 21.64%, and DEV10AR can improve by 23.12% over the MA baseline. These

Table 4.2: Estimated translation limits of scores achievable through re-ranking.

dataset	Up. BLEU	Low. BLEU	Up. NIST	Low. NIST
TST11AR	0.2916	0.1558	6.3467	4.8317
TST10AR	0.3013	0.1574	6.5602	5.1473
DEV10AR	0.3051	0.1692	6.6221	5.1339
TST10DE	0.2473	0.1513	6.5975	5.6003
DEV10DE	0.2553	0.1644	6.6549	5.7650

margins are statistically significant and would be considered excellent improvements if the system is able to achieve them. The TST10DE and DEV10DE datasets are not meant to test the possible improvement of the system here. It is meant to establish the baseline limits for the German-English system without recombination during *n*-best list generation. This is because the other methods do not add new hypotheses to the list, and the oracle re-ranking already selects the best existing hypothesis.

Table 4.3 shows the displacements of the datasets after oracle re-ranking. The absolute displacement is the number of swaps required to optimally rank all *n*-best lists in the corpus. The average normalized displacement for the corpora over all *n*-best lists is also displayed. The displacements suggest that the *n*-best lists are originally 25% away from

Table 4.3: Displacements caused by oracle re-ranking.

	Displacement	
	Absolute	Normalized
TST11AR	6,911,849	24.17%
TST10AR	1,853,701	22.67%
DEV10AR	5,257,817	28.51%

being optimal; however, the numbers may be misleading. Sentence-level BLEU may fail to re-rank a sentence when there are no *n*-grams co-occur in the reference and translation. The oracle uses in this research is BLEU-4 so if no 4-grams co-occur the sentence will not be re-ranked. All three datasets had at least three lists where the best hypothesis in last place. The displacement function also reveals that TST11AR had 444, TST10AR had 476, and DEV10AR had 192 *n*-best lists which were not re-ranked. Removing these sentences allows the average location of the hypothesis selected by the oracle to be calculated. The adjusted mean location is around the 45^{th} percentile.

4.3 Corpus-Level Re-Ranking Model Results

The corpus-level LM required the creation of a single LM for each of the corpora. The *n*-best lists contain words that were not translated out of the original Arabic. The words will not contribute to the quality of the translation. They are removed from the *n*-gram counts so that *n*-grams made of only words in English remain. The size of the LMs are displayed in Table 4.4. Declining weights in the counting function decreased the counts

Table 4.4: The *n*-gram counts of the corpus-level LM.

	Uniform Weight			Declining Weight		
	1-grams	2-grams	3-grams	1-grams	2-grams	3-grams
TST11AR	4022	40728	85800	4022	30836	59216
TST10AR	3888	38906	67918	3888	28888	43263
DEV10AR	3372	28953	56413	3372	22457	39706

of *n*-grams from those of the uniform weights. This is expected since the modified count discounts *n*-grams from lower hypotheses. Once the *n*-gram counts were combined and the models were generated, the system was trained.

The results of learning the correct feature weight are displayed in Figure 4.1. The behavior of the feature is similar in the three datasets. Toward the extrema, the scores are dominated by the new feature which reinforces that the global maximum exists between the selected training bounds. The score is above the NB baseline as the weight approaches one. This indicates that an absolute ordering based on the new feature would be beneficial. These results also indicate that the *n*-gram LM's feature weight, learned from one dataset, can be applied to others. The optimal feature weight was 0.17. At this weight, the system produced the results shown in Table 4.5 with the delta between the results and the NB baseline shown in parentheses.

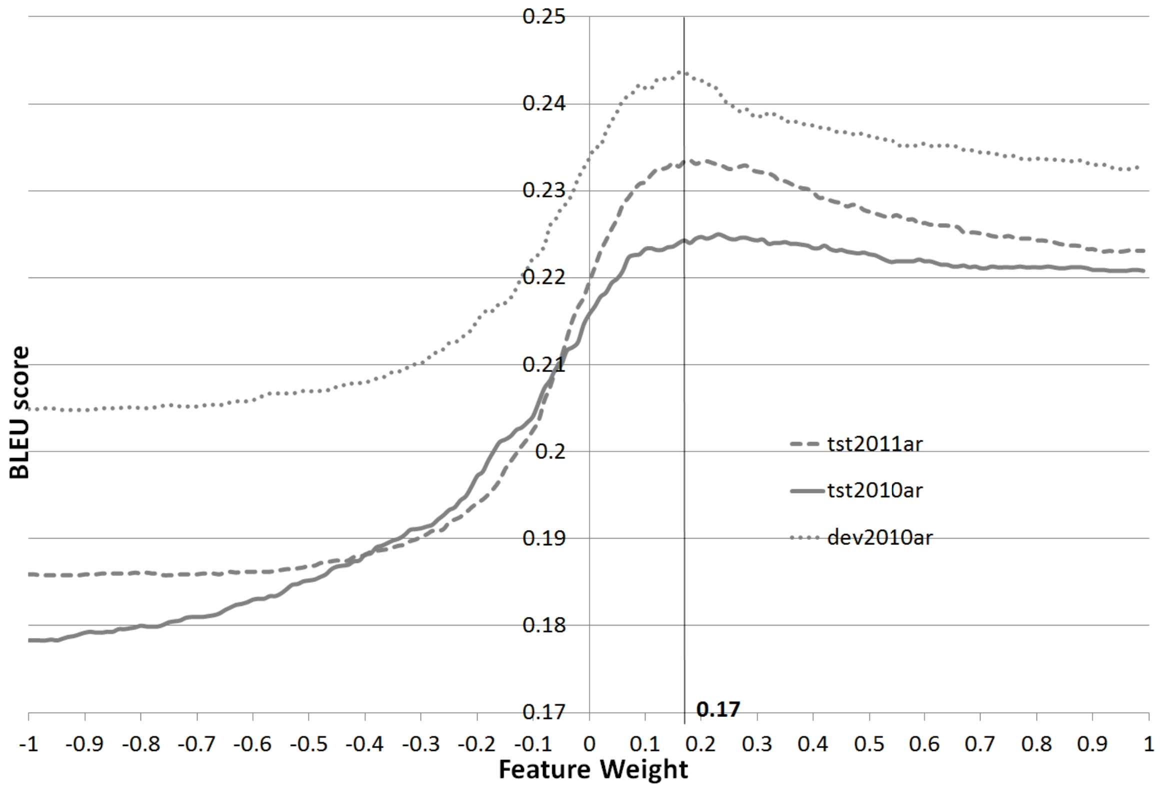


Figure 4.1: Translation score of datasets with different feature weights.

Table 4.5: Corpus-level re-ranking model results.

BLEU	TST11AR	TST10AR	DEV10AR
NB Baseline	0.2195	0.2146	0.2337
MA Baseline	0.2350	0.2477	0.2478
Uniform Weights	0.2334 (0.0139)	0.2243 (0.0097)	0.2436(0.0099)
Declining Weights	0.2319 (0.0124)	0.1955 (0.0094)	0.2440(0.0103)

NIST	TST11AR	TST10AR	DEV10AR
NB Baseline	5.5152	5.7684	5.8201
MA Baseline	5.7555	6.0184	6.0199
Uniform Weights	5.7961 (0.2578)	5.2920 (0.0284)	6.0271 (0.1947)
Declining Weights	5.7686 (0.2534)	5.3174 (0.0216)	6.0244 (0.2043)

In these results, there is an interesting trend. Uniform weights performed best for TST11AR and TST10AR while for DEV10AR there was more of an improvement when using declining weights. Because the differences between the uniform weight and the declining weight are small, it is likely due to standard error and cannot be extrapolated to make a general statement. The re-ranking showed an improvement of NIST scores for TST11AR and DEV10AR over those of the MITLL-AFRL final results. The differences in NIST and BLEU are their treatment of rare *n*-grams. The results indict that the models improved the accuracy of the rare *n*-grams even though the BLEU score decreased. All datasets showed an improvement over the BLEU for the *n*-best list base-line. If declining weight's performance increase is valid, then the results indicate that applying a better degradation model may improve the translation more. Additionally, using a better original ordering for the lists will likely result in Method 2 improving results further since the re-ranking receives the *n*-grams from the *n*-best lists present.

4.4 Producer-Level Re-Ranking Model Results.

The segmented feature sets created a language model for every speaker. The training data was therefore smaller resulting in less diverse *n*-grams. There were 76 LMs created for this approach. The average count of *n*-grams are displayed in Table 4.6. The translation evaluation results are displayed in Table 4.7 with the delta between the score and the NB baseline shown in parentheses.

Table 4.6: The *n*-gram counts of the corpus-level LM.

	Uniform Weights			Declining Weights		
	1-grams	2-grams	3-grams	1-grams	2-grams	3-grams
TST11AR	686	4060	6049	686	2969	4096
TST10AR	761	4366	5824	761	3137	3649
DEV10AR	785	5126	6345	785	3179	5166

Table 4.7: Producer-level re-ranking model results.

BLEU	TST11AR	TST10AR	DEV10AR
NB Baseline	0.2195	0.2146	0.2337
MA Baseline	0.2350	0.2477	0.2478
Uniform Weights	0.2306 (0.0107)	0.2241 (0.095)	0.2438 (0.0096)
Declining Weights	0.2302 (0.0096)	0.2227 (0.081)	0.2421 (0.0067)
NIST	TST11AR	TST10AR	DEV10AR
NB Baseline	5.5152	5.7684	5.8201
MA Baseline	5.7555	6.0184	6.0199
Uniform Weights	5.7265 (0.1909)	5.8153 (0.0469)	6.0113 (0.1661)
Declining Weights	5.7125 (0.1673)	5.8129 (0.0445)	5.9818 (0.1335)

The NIST and the BLEU scores were lower than the MA baseline for both methods of word counting and across all datasets. However, the improvements were not as substantial as those for the corpus-level model. The LMs are built from producer-level segments with far fewer sentences and so they are more sparse than the corpus-level models. The scores are improvements over the initial *n*-best list, which lends credibility to the hypothesis that producer-level re-ranking techniques are plausible. They may require larger producer samples such as the works of a popular columnist.

4.5 Local Context Re-Ranking Model Results

The local context language model is the most complicated to execute. The average number of monograms in the models was less than 100, and the average number of bigrams and trigrams in the models was around 200. It generates a LM for every sentence in the corpora. This takes more time and space than the other two methods. Table 4.8 shows the results from executing the local model re-ranking with a number of different weighting schemes with the delta between the score and the NB baseline shown in paretheses.

This method of re-ranking improved the translation of TST11AR and DEV10AR while reducing the translation quality of TST10AR. The results are improvements over the NB baseline; however, they are not better than the producer-level nor corpus-level model and so it is not the most viable approach of the three. Information can still be obtained from the results of the different weight array experiments. Weight arrays are referred to as W. A weight array is a set of values used to combine the *n*-gram counts from the individual *n*-best lists.

There are 5 weight arrays used in the test. W1 corresponds to the mask array $[1, 1, 1, 0, 1, 1, 1]$ and W2 represents the weight array of $[1, 1, 1, 1, 1, 1, 1]$. Both instances are *k*-windows of $k = 7$ because the cardinality of the arrays are 7. The results from these two weight arrays indicate that including the focus sentence in the model is neither beneficial nor detrimental to the quality of the translation. W3 shows the results from executing the local model with $k = 7$ with the weights set to $[2, 1.75, 1.5, 1, 1.5, 1.75, 2]$. W4 is a near inverse with the weights set to $[1, 1.5, 2, 1, 2, 1.5, 1]$. W5 tests the effects of extending k to be $k = 11$ with the mask array set to $[1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1]$. This is an extended version of W2 and so the scores are compared. The increase in k caused a slight increase in scores for TST11AR and DEV10AR and a decline in scores for TST10AR.

W4 achieved the highest average score of the weight arrays. This may indicate that sentences closer to the focus are more valuable to re-ranking than those further away; however, each dataset preferred different weight arrays. The margins are close enough together that it contradicts the hypothesis that one weight is better than the rest.

4.6 Recombination Results

The first recombination experiment uses the "-distinct-nbest" option to specify that Moses should generate unique hypotheses in the n-best list. It has the same effect as performing recombination with keep-best tie breaking. The results of the oracle re-ranking are shown in Table 4.9. It shows the upper and lower bound for achievable BLEU score for the translations of TST10DE and DEV10DE using $n = 200$. The BLEU and NIST scores for the initial translation are not different than the baseline in Table 4.1 because keep-best tie-breaking will not effect the top translation. There was a 10.1% increase in the max score of the BLEU in TST10DE and a 9.9% increase in DEV10DE. The lower bound estimated score dropped by 12.2% and 13.4% respectively. This indicates that the hypotheses that were prevented from being part of the n-best list are a distribution of better and worst hypotheses. This wider range of possible scores is provided without increasing the second-pass search space because n is unchanged.

If the system is time constrained, then it may not be an option to produce distinct hypotheses. The decoding for TST10DE and DEV10DE is run ten times, both with and without the distinct flag set. The execution time is recorded by a script that started recording just before the decoding launches and stops just after the decoding ends. The time for each run was then analyzed. The results are shown in Table 4.10. The tests were executed on the Ubuntu workstation while it was idle. The tests were run in a round-robin format. The four translations were decoded once before the test to stabilize memory management costs. Specifying the distinct option makes decoding take extra time. The translation time for TST10DE increasing 100.0% and the translation time for DEV10DE increased 104.9%. The results validate that time-constrained systems should avoid generating distinct *n*-best lists. However, if time is not a constraint the results show an increased in re-ranking potential that may be worth the extra time.

The second recombination technique deals with non-distinct *n*-best lists to reduce problem space. This is especially beneficial if the system cannot afford to generate distinct *n*-best lists due to time. Reducing the problem space may reduce run time. Table 4.11 contains the results of recombination with keep-best tie-breaking applied to the *n*-best lists from TST11AR, TST10AR, and DEV10AR. The average length of the *n*-best lists after recombination were just over half of the original length and within one point of the median value. The standard deviation was approximately half of the average value indicating that the distributions are relatively uniform. The distributions of the recombined lengths were uniform. One would expect the number of unique hypotheses to be linearly dependent on the number of words in the reference sentence. However, using the lengths of the reference sentences to form a sample covariance, the coefficients are -106, -46, and -109 making makes direct dependency highly unlikely.

Table 4.12 contains the results from the keep-worst recombination method as well as the averaging and the linear degradation combination. The delta between the score and the baseline are in parentheses. The alternate combination methods did not improve the translation. From this we can conclude that the metrics used during decoding are still effective heuristics of translation quality. Also, the number of paths leading to a given node does not mean that node is more likely. This makes sense as far as search trees are concerned validating our beliefs about the *n*-best lists . Interestingly, the linear degradation combination hurt the final score rather than help. This is likely because the absolute position of a hypothesis within the *n*-best list does not correlate directly to translation quality.

Calculating the displacement of the first two methods of recombination would yield no results since they do not actually re-rank the list. The displacement of TST11AR after averaging recombination is 6% and it is 48% after linear degradation combination. This would suggest that averaging has little effect on re-ranking nodes while linear degradation caused many small negative and positive changed due to the marginal decrease in translation quality across the sets.

Table 4.8: Local re-ranking model for masks with $k = 7$

BLEU Uniform Weights	TST11AR	TST10AR	DEV10AR
W1	0.2226 (0.0031)	0.2120 (-0.0026)	0.2362 (0.0025)
W2	0.2246 (0.0051)	0.2093 (-0.0053)	0.2390 (0.0053)
W3	0.2251 (0.0056)	0.2089 (-0.0057)	0.2382 (0.0045)
W4	0.2257 (0.0062)	0.2096 (-0.005)	0.2389 (0.0052)
W5	0.2234 (0.0039)	0.2107 (-0.0039)	0.2372 (0.0035)

BLEU Declining Weights	TST11AR	TST10AR	DEV10AR
W1	0.2212 (0.0017)	0.2135 (-0.0011)	0.2340 (0.0003)
W2	0.2249 (0.0054)	0.2099 (-0.0047)	0.2380 (0.0043)
W3	0.2247 (0.0052)	0.2089 (-0.0057)	0.2389 (0.0052)
W4	0.2253 (0.0058)	0.2086 (-0.0060)	0.2382 (0.0045)
W5	0.2219 (0.0024)	0.2111 (-0.0035)	0.2366 (0.0029)

NIST Uniform Weights	TST11AR	TST10AR	DEV10AR
W1	5.5368 (0.0216)	5.6704 (-0.098)	5.8341 (0.0140)
W2	5.5928 (0.0776)	5.6205 (-0.1479)	5.8984 (0.0783)
W3	5.5958 (0.0806)	5.6136 (-0.1548)	5.8885 (0.0684)
W4	5.6042 (0.089)	5.6237 (-0.1447)	5.8962 (0.0761)
W5	5.5619 (0.0467)	5.6516 (-0.1168)	5.8568 (0.0367)

NIST Declining Weights	TST11AR	TST10AR	DEV10AR
W1	5.5121 (-0.0031)	5.7007 (-0.0677)	5.7997 (-0.0204)
W2	5.5833 (0.0681)	5.6308 (-0.1376)	5.8749 (0.0548)
W3	5.5865 (0.0713)	5.6266 (-0.1418)	5.8815 (0.0614)
W4	5.5939 (0.0787)	5.6129 (-0.1555)	5.8777 (0.0576)
W5	5.5383 (0.0231)	5.6656 (-0.1028)	5.8342 (0.0141)

Table 4.9: Estimated translation limit score using BLEU oracle for distinct *n*-best lists.

dataset	Up. BLEU	Low. BLEU	Up. NIST	Low. NIST
TST10DE	0.2473	0.1513	6.5975	5.6003
DEV10DE	0.2553	0.1644	6.6549	5.7650
TST10DE distinct	0.2723	0.1328	6.8490	5.4986
DEV10DE distinct	0.2806	0.1424	6.9159	5.6385

Table 4.10: Timing of decoding.

Dataset	Avg. Time	Std. Dev.
TST10DE	88.07	1.43
DEV10DE	64.20	1.01
TST10DE D	175.78	1.32
DEV10DE D	131.54	1.50

Table 4.11: Recombination with keep-best tie-breaking.

	BLEU	NIST	Initial	Avg. final	Median	Std. Dev.
TST11AR	0.2195	5.5152	200	118.42	117	51.98
TST10AR	0.2146	5.7684	100	53.41	53	26.16
DEV10AR	0.2337	5.8201	200	108.35	107	49.31

Table 4.12: Recombination with alternate tie-breaking methods

BLEU	TST11AR	TST10AR	DEV10AR
Initial	0.2195	0.2146	0.2337
Keep-Worst	0.2088 (-0.0107)	0.2039 (-0.0107)	0.2260 (-0.0077)
Average	0.2046 (-0.0149)	0.2038 (-0.0108)	0.2199 (-0.0138)
Linear	0.2001 (-0.0194)	0.1976 (-0.0170)	0.2150 (-0.0187)

NIST	TST11AR	TST10AR	DEV10AR
Initial	5.5152	5.7684	5.8201
Keep-Worst	5.3481 (-0.1671)	5.5905 (-0.1779)	5.6736 (-0.1465)
Average	5.3225 (-0.1927)	5.5987 (-0.1697)	5.6326 (-0.1875)
Linear	5.2968 (-0.2184)	5.5296 (-0.2388)	5.6044 (-0.2157)

V. Conclusions

The need for quality, available, and inexpensive translations will increase as society continues to globalize. Statistical Machine Translation SMT provides a viable solution to the problem as the quality of the translations continue to improve. SMT offers a rich and deep field of study with many specialized areas where advances can be made toward this final goal. This research focused on improving SMT through *n*-best list re-ranking and re-ranking analysis.

5.1 Summary of Methodology

The research proposes a method of generating *n*-gram language models (LM) from the *n*-best lists generated during standard SMT decoding. The LMs were used to re-rank the *n*-best lists to produce a higher quality translation. In order to produce the LMs, two methods are presented to create *n*-gram counts from the *n*-best lists. The first method counts the *n*-gram occurrences in each hypothesis. This count is performed after like hypotheses are recombined. The second method discounts the occurrence of *n*-grams based on where they appear in the hypothesis. The second method showed improvement over the first method in some cases. This indicates that a varying degree of trust should be placed in the order of the *n*-best list. It is possible that this method of counting *n*-grams is more effective with higher quality starting *n*-grams.

The research presented three approaches to combining the *n*-gram counts of the *n*-best lists. The corpus-level approach combined the *n*-gram counts from all *n*-best lists within a corpus to form a re-ranking LM for that corpus. The approach was developed based on the hypothesis that a corpus has a domain and topic that dictate a semi-homogeneous style and word-choice. The producer-level approach grouped the *n*-gram counts based on their producer. This method is based on the theory that different producers have their own style communication. The local *k*-window method uses a window of *k* *n*-gram counts surrounding the focus list undergoing re-ranking. The window is combined using a weighting array to form an aggregate count for each *n*-best list in the corpus which is then turned into a language model. This method was based on the theory that near sentences will have *n*-gram co-occurrences.

5.2 Summary of Results

All levels of combination improved the translation over the NB baseline established by the default *n*-best list output by the Moses translation system with the exception of the TST10AR in the local *k*-window model. The local *k*-window LM did this using local *n*-gram context. Five ways of combining the neighboring *n*-gram counts were tested. The results indicate that the best method of combination varies with datasets. This improvements made by the *k*-window level were not as strong as the other two levels of combination. This indicates that the *n*-gram correlation in spatially related *n*-best lists is weak. The method of combining near sentence features may be applied to other features.

The producer-level combination improved translation scores more than the *n*-windows combination level. The main drawback to this method was the small sample size for each of the producers. It is possible that this method will be better for a larger corpora if the corpora contain large enough segments from producers. The larger samples will allow for larger language models with better representations of producer style and a richer collection of *n*-grams. This research suggests that a sample size of around 1600 sentences may be adequate as this was the size of the corpus-level combination.

The corpus-level model improved the translation the most. Its NIST scores were the only scores that improved more than the MA baseline established by the DoD-MITLL-AFRL team. The BLEU score did not overtake the MA baseline which indicates that the corpus-level *n*-gram occurrence model improved accuracy of rare *n*-gram co-occurrence. This result is interesting because the model contains many common *n*-grams. Further work should be conducted to determine if removing *n*-grams consisting of entirely common words such as “and the” would improve the re-ranking method.

A large benefit of this method of re-ranking is that it is not complex. The re-ranking is accomplished on a standard desktop computer and required less than 1 gigabyte of memory. The process also takes less than a minute to generate the models, re-rank the *n*-best lists, and evaluate the results of the three Arabic-Englishdatasets. These times are based on the code written in Python. If the methods are implemented in a compiled language such as C++, faster times can be expected. Another important observation is that the feature used in the re-ranking is not available during decoding and cannot be adapted for a one-pass system.

Additionally method of hypothesis recombination within the *n*-best lists were analyzed. By default, Moses-generated *n*-best lists are not recombined. The research has shown that the optimal method of recombining these hypotheses is the keep-best tie-breaking. Recombination did not hinder *n*-gram language model generation due to the lack of significant correlation between nodal density in the search frontier and node quality. This research shows that generating an *n*-best list with unique hypotheses will likely improve the maximum attainable translation score through re-ranking. The drawback to generating unique hypotheses is that it doubled translation time. For systems which require speed, recombining the *n*-best list after it is generated takes negligible time and decreases the problem space around 46% in the test datasets. This method of recombination does not improve the maximum achievable translation. Two-pass translation systems should apply one of these techniques to improve their speed or achievable quality of translation.

5.3 Future Work

During the course of the research, a number of topics for further research were apparent but could not be explored because they were outside the scope if this thesis. One topic is based on the work with *k*-window models. This method tested the co-occurrence of *n*-grams between nearby sentences. The key may not be *n*-gram co-occurrence but instead correlation. *n*-gram correlation is used in some sparse feature methods [41]. Instead of applying the model within sentences, the correlation would be between the words at the beginning and end of neighboring sentences. This feature could be used as a distance function between *n*-best list hypotheses. A Viterbi algorithm [38] could then be applied to the *n*-best lists.

Segmentation of corpora may provide improvements to translation. Subdivision relies on a feature of the source text that allows distinctions to be made and groups to be formed. In the research, these distinctions were used to group the corpus of *n*-best lists into producer groups. Instead of grouping unknown sentences, the same technique could be applied to training data. The distinction would not have to be producer, other features such as the average word fertility in the sentences could be used. It is only important that the feature can be calculated for the unknown sentences the system will translate. Each groups' corresponding know-text corpus would then be used to form a language model. During the second-pass, the group for each source sentence would be identified. The corresponding LM could them be applied to re-rank the list. Additionally, by separating these sentences, the system would be able to train a separate feature weight for each group.

While the results of the research indicate that there is promise in the proposed re-ranking process, further work should be done to test the methods on a wider range of datasets and over multiple optimization runs. Different models such as the corpus-level and producer-level LMs should be used simultaneously as features in a single re-ranking process. Finally, it would be beneficial to test the re-ranking process as a third-pass on two-pass systems such as the MITLL-AFRL system to determine if their translation quality can be increased further.

Appendix: Feature subset selection

A well tested approach to *n*-best list re-ranking discussed in Chapter 2 is to train the *n*-best list feature weights. The drawback was that over-fitting on the training data will usually lead to worse translation results. A hypothesis formed early in the research was that the re-ranking may be better if a subset of the initial features was used. This process cannot be performed without re-learning feature weights. The following information is the results of attempting to re-rank based on a subset of features without re-learning feature weights.

The re-ranking is performed by selecting a subset of the features and use the original weights. This is based on the hypothesis that certain features may be better removed than re-weighted. To test this hypothesis, feature removal combinations are tested through the standard log-linear equation. Instead of relearning feature weights, the weights from the initial decoding are masked against a vector of binary values which correspond to the selected features to keep and remove. There are eight features with non-zero values in the Arabic-English dataset, which would lead to 254 new combinations of metrics. A cursory set of seven combinations are tested. Each of the five non-zero phrase model features are used independently. In addition, the removal of the word penalty and the removal of the distortion penalty are tested.

Certain features may negatively affect the translation post-decoding. The feature weights are learned with the purpose of maximizing the outcome during decoding and theoretically the removal of one or more may improve the translation. For example, word penalty may not be as much of a concern since the hypothesis length is fixed by hypotheses in the *n*-best list. In this example, removing the length penalty is meant to remove the affect this feature has on the final ordering of the *n*-best list. In this experiment, each feature is tested for removal. The re-ranked corpus is saved to a file which is scored by mteval_v11b.

While the removing of certain features such as the LM is not expected to have a positive effect on the translation, it will be informative to see the corresponding changes in the translation quality.

Below in Table A.1 are the results from the selective feature re-ranking. The weight feature proved useful to prevent sentences from having appended phrases on them. For example the sentence. “While I am not fighting poverty I am fighting fires I am” may be chosen over the correct sentence “While I am not fighting poverty I am fighting fires”. The results are indicative that all metrics do in fact contain knowledge about the optimal translation. In some cases the local BLEU score did increase; however, as the data indicates none of the cases proved a full corpus translation improvement.

Table A.1: BLEU scores after re-ranking with feature subset.

	TM_1	TM_2	TM_3	TM_4	TM_5	LM	Rem. W	Rem. D
TST11AR	0.2186	0.1881	0.2002	0.1991	0.2061	0.1848	0.1848	0.2024
TST10AR	0.1900	0.2025	0.2011	0.2136	0.2125	0.2134	0.2193	0.2174
DEV10AR	0.2315	0.2046	0.2143	0.2131	0.2203	0.2009	0.2043	0.2186

		TST11AR	TST10AR	DEV10AR
NB Baseline	BLEU	0.2195	0.2146	0.2337
	NIST	5.5152	5.7684	5.8201
MA Baseline	BLEU	0.2350	0.2477	0.2478
	NIST	5.7555	6.0184	6.0199

TM_6 is missing because its final feature weight is 0 and it does not effect the ranking. We see that removing the reordering penalty has very little effect on the score. This may be indicative of a poor reordering model or that it is no longer needed. These results show

how certain weights and datasets have different levels of reaction to the removal of different features.

Bibliography

[1] "ARPA n-gram format". URL http://www.speech.sri.com/projects/srilm/manpages/ngram-format.5.html.

[2] "GNU Compiler Collection". URL http://gcc.gnu.org.

[3] "Google Hangouts". URL http://www.google.com/hangouts/.

[4] "Google Translate". URL translate.google.com/about/.

[5] "Microsoft". URL http://www.microsoft.com.

[6] "MT-Archive : Evaluation Publications since 2010". URL http://www.mt-archive.info/srch/eval-10.htm.

[7] "Oracle (Sun) Gride Engine". URL http://www.oracle.com/us/products/tools/oracle-grid-engine-075549.html.

[8] "Proficiency in the English language used for radiotelephony communications". URL http://cfapp.icao.int/fsix/lp/docs/resolution.pdf.

[9] "Python Programming Language". URL http://python.org.

[10] "Railteam - High speed network". URL http://www.railteam.eu/en/.

[11] "Scientific Linux Homepage". URL http://www.scientificlinux.org/.

[12] "SYSTRAN - Translation Technologies". URL http://www.systransoft.com/.

[13] "Ubuntu". URL http://www.ubuntu.com/.

[14] "United Nations - Official Languages". URL http://www.un.org/en/aboutun/languages.shtml.

[15] "World — Ethnologue". URL http://www.ethnologue.com/world.

[16] "Translation Bureau Benchmarking and Comparative Analysis Final Report". *Language Update*, 16(1), 2012. URL http://www.bt-tb.tpsgc-pwgsc.gc.ca/publications/documents/rapport-report-benchmarking-eng.pdf.

[17] AbuZeina, D. "Rescoring N-Best Hypotheses for Arabic Speech Recognition: A Syntax-Mining Approach". *. . . to Arabic Script- . . .*, 57–64, 2012. URL http://amta2012.amtaweb.org/AMTA2012Files/papers/wor-far-01.pdf#page=64.

[18] Al-Onaizan, Yaser, Jan Cuřín, Michael Jahr, Kevin Knight, John D. Lafferty, I. Dan Melamed, Franz-Josef Och, David Purdy, Noah A. Smith, and David Yarowsky. *Statistical Machine Translation*. Technical report, John Hopkins University Summer Workshop `http://www.clsp.jhu.edu/ws99/projects/mt/`, 1999.

[19] Aminzadeh, A Ryan, Tim Anderson, Ray Slyh, Brian Ore, Eric Hansen, Wade Shen, Jennifer Drexler, and Terry Gleason. "The MIT-LL/AFRL IWSLT-2011 MT System". *Proceedings of the eight International Workshop on Spoken Language Translation (IWSLT), San Francisco, CA*. 2011.

[20] Balakrishna, M. "N-best list reranking using higher level phonetic, lexical, syntactic and semantic knowledge sources". *Acoustics, Speech and* . . . , 413–416, 2006. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1660045.

[21] Black, Paul. "Merge Sort", August 2013. URL http://xlinux.nist.gov/dads//HTML/mergesort.html.

[22] Brown, P, J Cocke, and SD Pietra. "A statistical approach to language translation". *Proceedings of the 12th* . . . , 1988. URL http://dl.acm.org/citation.cfm?id=991651.

[23] Brown, Peter F., Stephen A. Della-Pietra, Vincent J. Della-Pietra, and Robert L. Mercer. "The Mathematics of Statistical Machine Translation". *Computational Linguistics*, 19(2):263–313, 1993.

[24] Brown, PF, J Cocke, and S Della Pietra. "A Statistical Approach to French/English Translation." *RIAO*, 1988. URL http://mt-archive.info/TMI-1988-Brown.pdf.

[25] Brown, PF, VJD Pietra, SAD Pietra, and RL Mercer. "The mathematics of statistical machine translation: Parameter estimation". *Computational linguistics*, 10598, 1993. URL http://dl.acm.org/citation.cfm?id=972474.

[26] Cettolo, Mauro and Marcello Federico. "Minimum Error Training of Log-Linear Translation Models". *Proc. of the International Workshop on Spoken Language Translation*, 103–106. Kyoto, Japan, 2004.

[27] Cettolo, Mauro, Christian Girardi, and Marcello Federico. "WIT^3: Web Inventory of Transcribed and Translated Talks". *Proceedings of the 16th Conference of the European Association for Machine Translation (EAMT)*, 261–268. Trento, Italy, May 2012.

[28] Chen, Boxing, Roldano Cattoni, Nicola Bertoldi, Mauro Cettolo, and Marcello Federico. "The ITC-irst SMT System for IWSLT-2005". *Proc. of the International Workshop on Spoken Language Translation*. October 2005.

[29] Chen, Stanley F. and Joshua Goodman. *An Emprirical Study of Smoothing Techniques for Language Modeling*. Technical Report TR-10-98, Computer Science Group, Harvard University, August 1998.

[30] Chiang, David, Kevin Knight, and Wei Wang. "11,001 New Features for Statistical Machine Translation". *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, 218–226. Association for Computational Linguistics, Boulder, Colorado, June 2009. URL http://www.aclweb.org/anthology/N/N09/N09-1025.

[31] Dahlgren, Peter. "The Internet, public spheres, and political communication: Dispersion and deliberation". *Political Communication*, 22(2):147–162, 2005.

[32] Dempster, Arthur P, Nan M Laird, and Donald B Rubin. "Maximum likelihood from incomplete data via the EM algorithm". *Journal of the Royal Statistical Society. Series B (Methodological)*, 1–38, 1977.

[33] Doddington, George. "Automatic evaluation of machine translation quality using n-gram co-occurrence statistics". *Proceedings of the second international conference on Human Language Technology Research*, 138–145. Morgan Kaufmann Publishers Inc., 2002.

[34] Doddington, George. "Automatic evaluation of machine translation quality using n-gram co-occurrence statistics". *Proceedings of the second international conference on Human Language Technology Research*, 138–145. Morgan Kaufmann Publishers Inc., 2002.

[35] Dugast, Loïc, Jean Senellart, and Philipp Koehn. "Statistical Post-Editing on SYSTRAN's Rule-Based Translation System". *Proceedings of the Second Workshop on Statistical Machine Translation*, 220–223. Association for Computational Linguistics, Prague, Czech Republic, June 2007. URL http://www.aclweb.org/anthology/W/W07/W07-0232.

[36] Duh, Kevin and Katrin Kirchhoff. "Beyond Log-Linear Models: Boosted Minimum Error Rate Training for N-best Re-ranking". *Proceedings of ACL-08: HLT, Short Papers*, 37–40. Association for Computational Linguistics, Columbus, Ohio, June 2008. URL http://www.aclweb.org/anthology/P/P08/P08-2010.

[37] Felice, Mariano and Lucia Specia. "Linguistic Features for Quality Estimation". *Proceedings of the Seventh Workshop on Statistical Machine Translation*, 93–100. Association for Computational Linguistics, Montreal, Canada, June 2012. URL http://www.aclweb.org/anthology/W12-3110.

[38] Forney Jr, G David. "The viterbi algorithm". *Proceedings of the IEEE*, 61(3):268–278, 1973.

[39] Graehl, Jonathan and Kevin Knight. "Training Tree Transducers". *Proceedings of the Joint Conference on Human Language Technologies and the Annual Meeting of the North American Chapter of the Association of Computational Linguistics (HLT-NAACL)*. 2004.

[40] He, Yifan, Yanjun Ma, Andy Way, and Josef van Genabith. “Rich Linguistic Features for Translation Memory-Inspired Consistent Translation”. *Proceedings of the 13th Machine Translation Summit (MT Summit XIII)*, 456–463. International Association for Machine Translation, 2011. URL http://www.mt-archive.info/MTS-2011-He.pdf.

[41] Hopkins, Mark and Jonathan May. “Tuning as Ranking”. *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, 1352–1362. Association for Computational Linguistics, Edinburgh, Scotland, UK., July 2011. URL http://www.aclweb.org/anthology/D11-1125.

[42] Hu, Xiaoguang, Haifeng Wang, and Hua Wu. “Using RBMT Systems to Produce Bilingual Corpus for SMT”. *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, 287–295. 2007. URL http://www.aclweb.org/anthology/D/D07/D07-1030.

[43] Hutchins, W. John. “Machine translation: a concise history”. *Computer aided translation: theory and practice*. 2007. URL http://www.hutchinsweb.me.uk/CUHK-2006.pdf.

[44] Kim, Seonho, Juntae Yoon, and Mansuk Song. “Structural Feature Selection For English-Korean Statistical Machine Translation”. *Proceedings of the International Conference on Computational Linguistics (COLING)*. 2000.

[45] Kneser, Reinhard and Hermann Ney. “Improved Backing-Off for M-Gram Language Modeling”. *Proceedings of the IEEE International Conference on Accoustics, Speech and Signal Processing*, volume 1. 1995.

[46] Knight, Kevin and Daniel Marcu. “Machine Translation in the Year 2004”. 965–968, 2005.

[47] Koehn, Philipp. “Europarl: A Parallel Corpus for Statistical Machine Translation”. *Proceedings of the Tenth Machine Translation Summit (MT Summit X)*. Phuket, Thailand, September 2005.

[48] Koehn, Philipp. *Statistical Machine Translation*. Cambridge University Press, 2010.

[49] Koehn, Philipp and Barry Haddow. “Towards Effective Use of Training Data in Statistical Machine Translation”. *Proceedings of the Seventh Workshop on Statistical Machine Translation*, 363–367. Association for Computational Linguistics, Montreal, Canada, June 2012. URL http://www.aclweb.org/anthology/W12-3144.

[50] Koehn, Philipp, Franz Josef Och, and Daniel Marcu. “Statistical Phrase Based Translation”. *Proceedings of the Joint Conference on Human Language Technologies and the Annual Meeting of the North American Chapter of the Association of Computational Linguistics (HLT-NAACL)*. 2003. URL http://acl.ldc.upenn.edu/N/N03/N03-1017.pdf.

[51] Li, Bo and Juan Liu. “Mining Chinese-English Parallel Corpora from the Web”. *Proceedings of the 3rd International Joint Conference on Natural Language Processing (IJCNLP)*. 2008.

[52] Liu, Ding and Daniel Gildea. “Syntactic Features for Evaluation of Machine Translation”. *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, 25–32. Association for Computational Linguistics, Ann Arbor, Michigan, June 2005. URL http://www.aclweb.org/anthology/W/W05/W05-0904.

[53] Martin, Joel, Rada Mihalcea, and Ted Pedersen. “Word Alignment for Languages with Scarce Resources”. *Proceedings of the ACL Workshop on Building and Using Parallel Texts*, 65–74. Association for Computational Linguistics, Ann Arbor, Michigan, June 2005. URL http://www.aclweb.org/anthology/W/W05/W05-0809.

[54] Nations, United. “CHARTER OF THE UNITED NATIONS”. *World Affairs*, 1945.

[55] Nelder, John A. and Roger Mead. “A simplex method for function minimization”. *Computing Journal*, 7(4):308–313, 1965.

[56] Och, Franz Josef. “Minimum Error Rate Training in Statistical Machine Translation”. Erhard Hinrichs and Dan Roth (editors), *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, 160–167. 2003. URL http://www.aclweb.org/anthology/P03-1021.pdf.

[57] Och, Franz Josef, Daniel Gildea, Sanjeev Khudanpur, Anoop Sarkar, Kenji Yamada, Alexander Fraser, Shankar Kumar, Libin Shen, David A. Smith, Katherine Eng, Viren Jain, Zhen Jin, and Dragomir Radev. “A Smorgasbord of Features for Statistical Machine Translation”. *Proceedings of the Joint Conference on Human Language Technologies and the Annual Meeting of the North American Chapter of the Association of Computational Linguistics (HLT-NAACL)*. 2004.

[58] Och, Franz Josef and Hermann Ney. “A Systematic Comparison of Various Statistical Alignment Models”. *Computational Linguistics*, 29(1), 2003.

[59] Och, Franz Josef, Nicola Ueffing, and Hermann Ney. “An Efficient A* Search Algorithm for Statistical Machine Translation”. *Workshop on Data-Driven Machine Translation at 39th Annual Meeting of the Association of Computational Linguistics (ACL)*. 2001.

[60] Papineni, Kishore, Salim Roukos, Todd Ward, and Wei-Jing Zhu. *BLEU: a Method for Automatic Evaluation of Machine Translation*. Technical Report RC22176(W0109-022), IBM Research Report, September 17 2001.

[61] Papineni, Kishore A. “Discriminative training via linear programming”. *Acoustics, Speech, and Signal Processing, 1999. Proceedings., 1999 IEEE International Conference on*, volume 2, 561–564. IEEE, 1999.

[62] Phillipson, R. *Linguistic Imperialism*. Oal Series. OUP Oxford, 1992. ISBN 9780194371469. URL http://books.google.com/books?id=4jVeGWtzQ1oC.

[63] Pierce, John R. and John B. Carroll. *Languages and Machines — Computers in Translation and Linguistics*. Technical report, Automatic Language Processing Advisory Committee (ALPAC), National Academy of Sciences, 1966. URL http://www.mt-archive.info/ALPAC-1966.pdf.

[64] Press, WH, SA Teukolsky, WT Vetterling, and BP Flannery. "Numerical recipes in C++. 2002". *Cambridge UP*.

[65] Rafalovitch, Alexandre and Robert Dale. "United Nations General Assembly Resolutions: A Six-Language Parallel Corpus". *Proceedings of the Twelfth Machine Translation Summit (MT Summit XII)*. International Association for Machine Translation, 2009.

[66] Rayner, M, D Carter, V Digalakis, and P Price. "Combining knowledge sources to reorder n-best speech hypothesis lists". *Proceedings of the workshop ...*, 1994. URL http://dl.acm.org/citation.cfm?id=1075858.

[67] Shannon, Claude Elwood. "A mathematical theory of communication". *ACM SIGMOBILE Mobile Computing and Communications Review*, 5(1):3–55, 2001.

[68] Simard, Michel, Cyril Goutte, and Pierre Isabelle. "Statistical Phrase-Based Post-Editing". *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*, 508–515. Association for Computational Linguistics, Rochester, New York, April 2007. URL http://www.aclweb.org/anthology/N/N07/N07-1064.

[69] Sokolov, A, Guillaume Wisniewski, and F Yvon. "Non-linear n-best List Reranking with Few Features". *cl.uni-heidelberg.de*, 2002. URL http://www.cl.uni-heidelberg.de/~sokolov/pubs/sokolov12nonlinear.pdf.

[70] statmt.org. "Moses - Main/Homepage", 2014. URL http://www.statmt.org/moses/.

[71] Stolke, Andreas. "SRILM - An Extensible Language Modeling Toolkit". *Proceedings of the International Conference on Spoken Language Processing*. 2002.

[72] Tillmann, Christoph, Stephan Vogel, Hermann Ney, and Alex Zubiaga. "A DP-based Search Using Monotone Alignments in Statistical Translation". *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics (ACL)*. 1997.

[73] Utiyama, Masao, Daisuke Kawahara, Keiji Yasuda, and Eiichiro Sumita. "Mining Parallel Texts from Mixed-Language Web Pages". *Proceedings of the Twelfth Machine Translation Summit (MT Summit XII)*. International Association for Machine Translation, 2009.

[74] Varga, Dániel, Péter Halaácsy, András Kornai, Voktor Nagy, László Németh, and Viktor Trón. “Parallel corpora for medium density languages”. *Proceedings of the RANLP 2005 Conference*, 590–596. 2005.

[75] Vilar, David, Daniel Stein, and Hermann Ney. “Analysing soft syntax features and heuristics for hierarchical phrase based machine translation”. *Proceedings of the International Workshop on Spoken Language Translation (IWSLT)*, 190–197. 2008.

[76] Watanabe, Taro and Eiichiro Sumita. “Bidirectional Decoding for Statistical Machine Translation”. *Proceedings of the International Conference on Computational Linguistics (COLING)*. 2002.

[77] Weaver, Warren. “Translation”. *Machine translation of languages*, 14:15–23, 1955.

[78] Winograd, Terry A and Carlos Fernando Flores. *Understanding computers and cognition: A new foundation for design*. Intellect Books, 1986.

[79] Xiang, Bing, Niyu Ge, and Abraham Ittycheriah. “Improving Reordering for Statistical Machine Translation with Smoothed Priors and Syntactic Features”. *Proceedings of Fifth Workshop on Syntax, Semantics and Structure in Statistical Translation*, 61–69. Association for Computational Linguistics, Portland, Oregon, USA, June 2011. URL http://www.aclweb.org/anthology/W11-1007.

[80] Zens, Richard, Franz Josef Och, and Hermann Ney. “Phrase-Based Statistical Machine Translation”. *Proceedings of the German Conference on Artificial Intelligence (KI 2002)*. 2002. URL www.rzens.com/Zens_KI_2002.pdf.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704–0188

The public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden to Department of Defense, Washington Headquarters Services, Directorate for Information Operations and Reports (0704–0188), 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202–4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to any penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number. **PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.**

1. REPORT DATE *(DD–MM–YYYY)* 27–03–2014	**2. REPORT TYPE** Master's Thesis	**3. DATES COVERED** *(From — To)* Oct 2013–Mar 2014

4. TITLE AND SUBTITLE Improving Statistical Machine Translation Through N-best List Re-ranking and Optimization	**5a. CONTRACT NUMBER**
	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S) Keefer, Jordan S., Second Lieutenant, USAF	**5d. PROJECT NUMBER**
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Air Force Institute of Technology Graduate School of Engineering and Management (AFIT/EN) 2950 Hobson Way WPAFB, OH 45433-7765	**8. PERFORMING ORGANIZATION REPORT NUMBER** AFIT-ENG-14-M-43
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) Intentionally left blank.	**10. SPONSOR/MONITOR'S ACRONYM(S)**
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEMENT

DISTRIBUTION STATEMENT A:
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

13. SUPPLEMENTARY NOTES

This work is declared a work of the U.S. Government and is not subject to copyright protection in the United States.

14. ABSTRACT

Statistical machine translation (SMT) is a method of translating from one natural language (NL) to another using statistical models generated from examples of the NLs. The quality of translation generated by SMT systems is competitive with other premiere machine translation (MT) systems and more improvements can be made. This thesis focuses on improving the quality of translation by re-ranking the n-best lists that are generated by modern phrase-based SMT systems. The n-best lists represent the n most likely translations of a sentence. The research establishes upper and lower limits of the translation quality achievable through re-ranking. Three methods of generating an n-gram language model (LM) from the n-best lists are proposed. Applying the LMs to re-ranking the n-best lists results in improvements of up to six percent in the Bi-Lingual Evaluation Understudy (BLEU) score of the translation.

15. SUBJECT TERMS

Natural Language Processing; N-best List Re-ranking; Statistical Machine Translation; Contextual Feature Selection

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON**
a. REPORT	**b. ABSTRACT**	**c. THIS PAGE**			Maj Kennard R. Laviers (ENG)
U	U	U	UU	89	**19b. TELEPHONE NUMBER** *(include area code)* (937) 255-3636 x0000 kennard.laviers@afit.edu